NETWORK CODING FOR DISTRIBUTED STORAGE NETWORKS

BY

AHMED A. AL-HABOB

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

In

TELECOMMUNICATION ENGINEERING

DECEMBER 2015

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **AHMED A. AL-HABOB** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN TELECOMMUNICATION ENGINEERING.**

Thesis Committee

Dr. Sameh Sorour (Adviser)

Dr. Ashraf Mahmoud (Member)

Dr. Wessam Mesbah (Member)

Dr. Ali A. Al-Shaikhi
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

30/12/15
Date

Dedication

To my parents, wife, son, sisters, and brothers for their endless support and love.

ACKNOWLEDGMENTS

All praise and thanks be to Almighty Allah, the one and only who helps us in every aspect of our lives.

Acknowledgment is due to King Fahd University of Petroleum and Minerals for giving me this precious opportunity to resume my Master degree.

I would like to express deep gratefulness and appreciation to my Thesis advisor Dr. Sameh Ossama Sorour for his continuous help, guidance, and encouragement throughout the course of this work. He spent a lot of his precious time helping me and advising me at each step. Beside my advisor, I would like also to thank my Thesis committee members: Dr. Ashraf S. Hasan Mahmoud and Dr. Wessam Mesbah for their great help and cooperation, which contributed significantly to the improvement of this work.

Next, I would like to pay my humble respect and the deepest thanks from my heart to my parents. I am extremely grateful to them to have sacrificed themselves to give me the finest education, the warmest care, the righteous upbringing and the best in everything.

My final warmest thanks go to my wife and my son Yahya, the beautiful smiles in my life, to whom I owe the renewal of my strength to accomplish this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

IDNC	Instantly Decodable Network Coding
LF	Lossy Feedback
ML	Maximum Likelihood
DSNs	Distributed Storage Networks
EC	Erasure Coding
NC	Network Coding
FNC	Full Network Coding
ONC	Opportunistic Network Coding
PMP	Point-to-Multipoint system
MMP	Multipoint-to-Multipoint system
SSP	Stochastic Shortest Path
POSSP	Partially Observable Stochastic Shortest Path

SRC	Simple Regenerating Codes
DTU	Download Time Unit
R	Repetition index of the stored files
FVE	Full Vertex Elimination
CB	Client Block
$\mathbf{N_c}$	Number of Clients
$\mathbf{N_f}$	Number of Files
$\mathbf{N_s}$	Number of Servers
$\mathbf{P_{ij}}$	File corruption probability on the down-link sub-channel
$\mathbf{q_{ij}}$	File corruption probability on the up-link sub-channel

THESIS ABSTRACT

NAME:	Ahmed A. Al-habob
TITLE OF STUDY:	Network Coding For Distributed Storage Networks
MAJOR FIELD:	Telecommunication Engineering
DATE OF DEGREE:	December 2015

In this thesis work, we study the problem of reducing the file download completion delay from multiple servers system. This is achieved by utilizing conflict-free Instantly Decodable Network Coding (IDNC). IDNC can speed up the download process by taking advantage of the clients' side information. At each time epoch, a client can tune to only one server. However, more than one server will target the same client simultaneously causing transmission conflicts. To accomplish conflict-free download mechanisms, a dual conflict IDNC graph model is proposed. This model considers the transmission conflicts among the servers in finding the optimal file combinations. Using this model, we design a channel-aware heuristic algorithm which is used in solving the download time minimization problem in such systems. The performance of the proposed algorithm (in terms of the completion delay) is compared to that of the conventional separated IDNC scheme in

which every server tries to minimize its own completion delay without considering the transmissions of other servers. The proposed scheme is shown to achieve significant reduction in completion delay as compared to the separated IDNC scheme. Both lower and upper bounds of the performance of conflict-free IDNC algorithm is derived for the scenario of one file request per client. The proposed heuristic algorithm shows near optimum performance compared to the optimum solution acquired using Bron-Kerbosch algorithm. Furthermore, a lossy feedback (LF) environment is investigated (i.e., when feedback loss event occurs). Under such an environment, the uncertainty about file reception at the targeted client would drive the server to carry out partially blind file combinations selection. To find efficient selection policy that reduces the completion delay of the conflict-free IDNC in such environments, the server performs maximum likelihood (ML) state estimation to evaluate the system state and then updates the dual conflict IDNC graph accordingly.

ملخص الأطروحة

في هذة الأطروحة قمنا بدراسة مسألة تقليل زمن تحميل الملفات الموزعة على خادمات بيانات متعددة. أرتكزت هذة الطرق على الإستفادة من نظرية ترميز الشبكات الأني الخالي من التضارب (IDNC). يتميز هذا النوع من الترميز بقدرتة على التسريع من زمن معالجة تحميل الملفات عن طريق الإستفادة من المعلومات الجانبية الخاصة بالعميل المرتبط بالخادمات المتعددة. على الرغم من انة في كل فترة زمنية كل عميل يتم ربطة مع خادم شبكة واحد فقط؛ أكثر من خادم للشبكة سوف يقوم بإستهداف نفس العميل مما يتسبب لة تضارب في عملية الإرسال. للقيام بإنجاز عملية تحميل البيانات بدون أي تضارب؛ قمنا بإقتراح خوارزمية رسم بياني مزدوجة التضارب ل(IDNC). يقوم هذا النموذج المقترح بإعتبار تضارب التراسل بين الخادمات المختلفة كمعلومات مفيدة لإختيار الأجزاء المثلى من الملف ليتم تحميلها للعميل الطرفي. بواسطة هذا النموذج؛ قمنا بتصميم خوارزمية إسترشادية تقوم بتحسس حالة قنوات الإتصال بين العميل وباقي خادمات الشبكة وتستخدم هذا المعلومات لتقليل الزمن اللازم لتحميل الملفات في هذة الشبكة. تمت مقارنة الأداء العام للخوارزمية المقترحة من جهة تقليل زمن التأخير في تحميل البيانات مع نموذج الشبكة الإعتيادية حيث الخادمات المتعددة تقوم بتقليل زمن تحميل البيانات الخاصة بها بدون إستشعار الخادمات الأخرى المرتبة معها بنفس الشبكة. تم إنجاز تقليل ملحوظ في زمن التحميل بإستخدام الخوارزمية المقترحة مقارنة نموذج الشبكة الإعتيادية. قمنا بإشتقاق حد ادنى وحد اعلى لأداء الخوارزمية المقترحة و كان أدائها قريب من الحل الأمثل المحصل علية بإستخدام خوارزمية Bron-Kerbosch. إضافةً إلى ذلك؛ تم دراسة أداء النموذج المقترح في وجود بيئة من قنوات الاتصال غير مثالية بين العميل و الخادم. في مثل هذة الحالات تحدث عمليات تضارب في الشبة والتي يتم تغذيتها عكسياً لخادمات الشبكة. نتيجة لذالك؛ فإن مقدار الشك عن حدوث إستلام للملف بواسطة العميل المستهدف قد تدفع بخادم الشبكة بأن يقوم بإعتماد نموذج التحميل الإعتيادي بدون أخذ الخادمات المجاورة في الإعتبار. للقيام بعملية كفؤة لإختيار الملفات المرسلة وتقليل التضارب في مثل هذة البيئة؛ يقوم خادم الشبكة بإستخدام طريقة التشابة الأكبر (ML) للحصول على أفضل تقدير للحالة الحالية للشبكة وللقيام بتحديث الرسم البياني للشبكة على حسب هذة القراءة.

CHAPTER 1

INTRODUCTION

In the last decade, an exponential growth of storage systems capacity has been witnessed as the need for storing e-mails, photos and videos has increased. This growth encouraged the development of new data storage techniques in order to enhance the two major performance indicators of data storage systems, namely, reliability and availability [1, 2]. To this end, new research field has emerged known as distributed storage networks (DSNs) coding. In which, distributed storage over multiple storage nodes with an appropriate content repetition was considered as the easiest and most potential solution to improve the availability and reliability of storage systems [1, 3]. One storage technique is full replication of the same data in multiple storage units, which imposes extremely high storage overhead. Another storage technique that can provide better redundancy-reliability trade-off is Erasure Coding (EC) (a scheme designed to protect data against partial data loss events). In erasure coding, the data file is divided into file chunks, encoded and expanded with some redundant data, which are stored among differ-

ent storage nodes. If data become corrupted at some storage nodes, the system would be able to extract the damaged data from the information about it stored elsewhere in the system [3, 4, 5]. On the other hand, DSNs have also attracted more interest lately to improve system availability and to efficiently schedule the process of downloading data from these storage nodes (servers) to nodes (clients) demand files. With the emergence of smart phones, the 4G and 5G cellular network architectures, the data download problem has become more attractive in the wireless scenarios. Consequently, the data download problem was mutated from a conventional full package download problem into a download completion problem [6, 7]. The huge storage capabilities of modern client equipment enable these devices to benefit from the already downloaded data pieces and request only the missed ones. These advances in client equipment gave birth to a new scheduling and routing scheme called Network Coding (NC). Network coding is the concept of packets blending at the transmitter or the intermediate points in order to attain uttermost information usefulness of each encoded packet [8, 9, 10]. The works in [1, 2, 11] introduced network coding as an intelligent file storage method with convenient redundancy. By storing the data in a proper way with some acceptable redundancy, the system can retrieve the content of a failing node from the other nodes. In [2], a simple regenerating coding method was developed for DSNs. In this coding method the system is able to retrieve missing files stored over multiple storage servers. A tree-structured connectivity model consists of a single queue connected to a set of storage servers, that receives and schedule user

requests for content was proposed in [11]. However, this approach did not take into account two practical characteristics of DSNs. First, the DSNs can operate in multipoint-to-multipoint system model. The multipoint-to-multipoint DSNs model is attractive in many applications due to the remarkable provided higher system availability and the reduction of the client-block events probabilities. Second, some of the storage files have already been downloaded by the clients. To minimize the time required to download other files, these downloaded files can be utilized as clients side information. Moreover, file transmission from servers in both wireless and wired networks suffer from channel impairments which lead to file corruption. File corruption introduces an additional downloading overhead in order to request for corrupted files re-transmission.

Another two main network coding approaches have been investigated in the literature, namely the full network coding (FNC) approach [12, 13] and the opportunistic network coding (ONC) approach [14, 15]. In FNC, all the packets are mixed together with deterministic or random non-zero finite field coefficients. Consequently, a client should manipulate all the packets of a certain frame before starting to decode. On the other hand, ONC utilizes the knowledge of previously downloaded packets at each client to efficiently select the encoded packet combinations that would ensure the decodability of encoded packets after each transmission for all clients or a subset of them. A type of opportunistic network coding named instantly decodable network coding (IDNC) in which the transmitted coded packet is decodable at the same instant. In IDNC, the low decoding

delay and no buffering requirement are considered as the most appealing properties which are applicable for a wider range of real-time transmissions [16, 17]. Moreover, IDNC encoding can be implemented using simple equipment due to the fact that the encoding and the decoding processes are binary XOR operations only [18, 19, 17].

The completion time minimization problem for point-to-multipoint (PMP) networks using IDNC assuming erasure channels on the forward links from the transmitter to the receivers was considered in [20, 21]. The results showed that the best scheduling scheme is the one that gives higher chance to those receivers that require more packets and suffers from worst channel conditions. Another major extension to the IDNC completion time problem was proposed in [22, 23], where feedback loss events involved in the PMP system model are considered. Our problem differs from this (conventional) IDNC completion time minimization problem in that multiple transmitters transmit at the same time. This introduces an additional scheduling challenge to avoid targeting the same receiver by more than one transmitter at the same time epoch. To the best of our knowledge, this work introduces a novel conflict-free IDNC algorithm suitable for distributed storage systems (or any multipoint-to-multipoint system) involving simultaneous transmissions from multiple transmitters. Consequently, this paradigm triggered the following essential question: *In DSNs, can we upgrade the point-to-multipoint (PMP) IDNC network coding to be conflict-free IDNC when applied in multipoint-to-multipoint (MMP) systems with perfect, erasure and lossy feedback channels?*

To answer the aforementioned question, a novel conflict-free IDNC algorithm that can be applied to MMP systems that involve simultaneous transmissions is developed. Our algorithm aims at both reducing the completion time and ensuring conflict-free transmissions under perfect or erasure channel conditions. To further reduce complexity, we devise a channel-aware heuristic algorithm to find the maximum weighted independent set in the dual conflict IDNC graph. We compare the performance of our proposed dual conflict IDNC algorithm and conventional IDNC. We extend the study of the completion time minimization problem of conflict-free IDNC to the lossy feedback (LF) environment. The stochastic shortest path (SSP) problem [1] in perfect feedback was extended to partially observable SSP (POSSP) formulation, reflecting the uncertainties resulting from unheard feedback events. This formulation is then utilized to identify the maximum likelihood (ML) state of the system in events of unheard feedback, which is then employed to determine a partially blind dual conflict IDNC graph update. We derive approximation for the chromatic number [2] of the dual conflict IDNC graph assuming systematic fixed file placement is utilized at the servers, perfect channel conditions, and that each client wants only one file. Thus we derived a lower and an upper bounds of the conflict-free IDNC algorithm performance in this special case.

[1] SSP problem is a class of probabilistic planning problems which describe a wide range of possible scenarios where the purpose of the active agent is to reach a goal state in the minimum costly way from any non-goal state using actions with probabilistic outcomes.

[2] chromatic number is the smallest needed labels or colors required to color the vertices of graph so that any two adjacent vertices do not share the same colour.

1.1 Introduction to Distributed Storage Networks DSNs

The last decade has seen an exponential growth of the need to scalable, reliable and continuously operational data storage systems as the need for storing the E-mails, YouTube videos, Facebook notes, Instagram pictures, Dropbox shared files and so on, increased. Storage system scalability implies that the system will remain working efficiently with the significant increase in the number of users connected to the system. In the other hand, the system has the ability to optimize the required resources to fit the reduction in the number of the users [24]. Another desired property of a storage system is reliability, storage system reliability is a measure of the system ability to overcome the unpredictable catastrophes [25]. Distributed Storage Network (DSN) (a.k.a. cloud storage system) is considered as the easiest and the most interesting solution to reliably store and retrieve data. DSN is a storage system where the data stored over multiple servers with proper redundancy [26]. Distributing the content over multiple servers enhances the scalability; the system controller can add more servers to fit the increasing number of users. The benefits of distributing the data over multiple servers, of course, do not come without cost (the overheads of creating, maintaining and updating the servers content). Another major challenge distributed storage networks system has to address is skewing in the data load, which can either be in the distribution of data items or data access over the servers in the system [27]. DSN is a high-speed network whose primary purpose is the transfer of data between storage elements.

A DSN consists of a set of (1) storage nodes or servers located in different locations, (2) communication infrastructures, which provides physical connections among these servers and also between the servers and the clients and (3) data placement controller, which includes a management layer to organize the storage servers, the connections and distribute the data among the servers [28]. In modern wireless sensor networks and applications based on sensor networks, obtaining incessant and reliable storage over distributed unreliable nodes is a desirable option for robust data recovery, especially in emergency or catastrophic scenarios [4, 5]. The most common model by which data and files are transferred on the Internet is the client-server model. A distributed server transmits the file to each client that requests it. The clients usually speak to the server, not to each other. The major merits of this model are that it is simple to set up and implement and the files are always available since the servers will be dedicated to serve the clients, therefore the clients are always on and connected to the Internet. Applications involve peer-to-peer storage systems and storage in large data centers such as TotalRecall and OceanStore use nodes distributed across the Internet for distributed file storage.

1.1.1 Servers content replication

One crucial factor in any storage system is replication offset. Full redundant is the simplest replication scheme in which the the content of a server replicates into other redundant servers. In addition to the simplicity, full redundant scheme achieved a remarkable performance in optimizing the load balancing dilemma and

also its disaster recovery capability depends on the number of the replicas [29]. RAID (Redundant Array of Independent Disks; originally inexpensive disks) is an application that adopts full redundant scheme. In RAID, identical content is repeated in multiple storage disks. Since multiple disks enhances fault tolerance, storing data redundantly also increases the mean time between failures [30]. The full redundant scenario (all the nodes store the same data with one hundred percent repetition) is the safest option but its cost is very high. The literature in [1, 2, 11] introduce the network coding as an intelligent file (or file chunks) storing method with acceptable redundancy. By storing the data in a proper way with some acceptable redundancy we can retrieve the content of a failed server from the other servers. The other servers can retrieve the data by downloading a convenient subset of the encoded files, so that this subset is sufficient to retrieve the lost data. The most popular encoding method is *erasure coding*, which will be discussed in the following section.

1.1.2 Erasure Coding (EC)

Erasure coding is an algorithm that allows the retrieval of destroyed (missing) data pieces from a set of original (survived) data pieces. Erasure coding is a generalization of replication that does not require to repeat all the server content and it achieves a remarkable redundancy-reliability trade-off compared to the classical replication-based systems. Erasure coding algorithm runs to generate additional encoded data pieces and these encoded data pieces scattered among the servers

[3]. Erasure coding provides much higher disaster recovery capability compared to full replication scheme, however, it introduces higher computational complexity [3, 2]. To enable the missed files retrieval, different erasure coding algorithms were developed (refer to [1, 2] and references therein). The following equation: $n = k + m$ represents the recovery tolerance offered by erasure coding. Where k is the number of original files, m stands for the redundant or the additional files that are embedded to provide immunity against unexpected failures and the overall number of files resulted by the erasure coding process is denoted by n. A subclass of the EC is the Simple Regenerating Codes (SRC) proposed in [2]. In SRC the server content divided into equally sized files and the encoding process can be implemented using binary XOR. Binary XOR encoding method has attractive properties such as implementation simplicity and the slight encoding-decoding delay. To show the retrieval process of the failed server content using SRC scheme, let us consider the following simple example:

A distributed storage system with 4 storage servers each server stores 2 files, the content of servers is as follows:

s_1 stores $\{f_1, f_2\}$.

s_2 stores $\{f_3, f_4\}$.

s_3 stores $\{f_5, f_6\}$.

s_4 stores $\{f_7, f_8\}$.

Once any server failed, the system can not retrieve the lost data stored in that server. To make this system immune against one server fail disaster, additional

file storage capacity will be embedded in the storage capacity of the servers. Each server will be of size 3 files and the new added file slots will stored the encoded files as follows:

s_1 stores $\{f_1, f_2, f_4 \oplus f_5\}$.

s_2 stores $\{f_3, f_4, f_6 \oplus f_7\}$.

s_3 stores $\{f_5, f_6, f_1 \oplus f_8\}$.

s_4 stores $\{f_7, f_8, f_2 \oplus f_3\}$.

The resulted system has the ability to retrieve the content of any failed server. When the content of a single server is lost due to unpredictable catastrophe, the system controller starts the recovery process by decoding the information stored in the survived servers.

Fig. 1.1 illustrates the system with Server 2 out of service and the retrieval process of Server 2 content. To ensure uninterrupted system availability the retrieved information can be stored temporary in another backup server until the failed server be maintained. In this example, the extra storage space required to enable the system to recover one server failure disasters is 4 file slots. On the other hand, the full redundant scheme required 8 extra slots to achieve the same recovery capability.

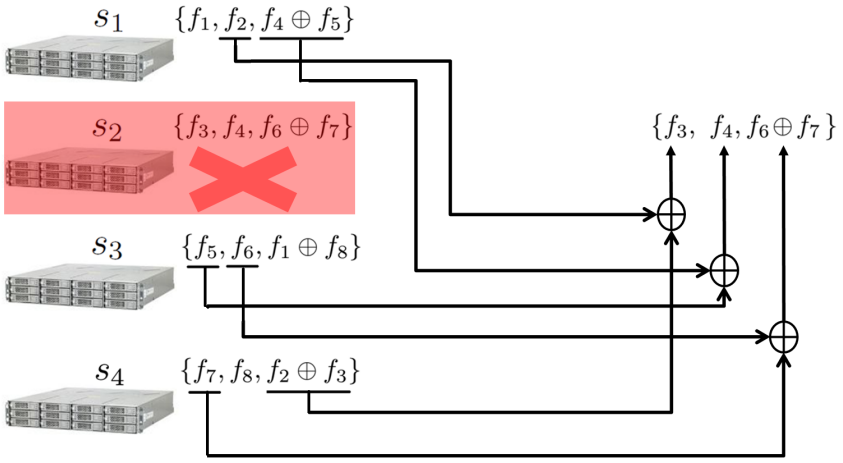


Figure 1.1: The repair of Sever 2 using simple regenerating codes.

A systematic approach for placement of the original and encoded files and also the encoding approach was introduced in [2], the authors proved that the coding rate (space efficiency) of the SRC which equals to the ratio of the storage space required to store the original (uncoded) information, to the total storage space required approaches $\frac{2}{3}$, and the server failure tolerance is one fourth the total number of servers.

1.2 Introduction to Network Coding (NC)

Communication networks are a predominant component of the modern life. Everywhere around us, machines and people exchange information at different scales, speeds and throughputs. The network is everywhere: from small number of connected devices such as Bluetooth networks, which are short-range wireless interconnection of computers, cellular phones and other electronic devices, passing by all voice communications that the local phone company offers and all the mobile communication technologies and standards to the wide expanse of connectedness rely on TCP/IP protocols, that literally make the communication networks out of thin air. Regarding the design of communication networks, one fundamental issue should be taken into account, how network traffic is delivered between network terminals. This crucial issue will identify the utilizing efficiency of the network resources. Modern Communication network topologies allow independent data stream to share the same network resources. The transmission nature of wireless networks requires simultaneous transmissions which typically result in some packets to be lost, therefore the transmitter requires to re-transmit the missed packets. This paradigm has initiated a new area of research, namely *network coding*. Network coding (NC) is a new technique in communication networks, where the transmitter or the intermediate nodes are allowed to combine (code) the data and perform coding operations on the contents of the packets, normally without changing the packet size to maximize the information utility of each encoded packet. Network Coding attracts the attention of many researchers as a scheduling

and routing scheme that can achieve remarkable enhancement in the information flow with different network transmission modes and topologies [8, 9, 31]. Network coding allows the transmitter or the intermediate nodes to perform algebraic operations inside the network. The algebraic operations and the mixture of data lead to more secure data exchange. In network coding, the transmitted data does not depend only on single packet but also on the contents of other packets that happen to be mixed together and sharing the same channel/route at the same time of transmission. For this reason, communication networks with network coding is more secure and more immune against hacking, eavesdropping and other forms of attack than networks with traditional data transmission. In the literature, two main network coding approaches can be considered, namely Full Network Coding (FNC) [12, 13] and Opportunistic Network Coding (ONC) [14, 15]. In the FNC approach, the wanted packets are coded all together with random or deterministic coefficients. Consequently, in FNC all the wanted packets are encoded and transmitted to the receivers, this leads to high computation complexity. On the other hand, the feedback-based or the ONC approach takes advantage of the receivers side information (the existed and the wanted packets) and then carefully select the suitable packets combination to insure the decodability of encoded packet in each encoded transmission for all the receivers or a subset of them.

1.2.1 Instantly Decodable Network Coding (IDNC)

IDNC is a type of the feedback-base or the ONC, in which the encoded IDNC packets are selected to be decodable at the same instant by a subset of the receivers (or hopefully all the receivers) with out the need to be stored for future decoding. IDNC has attractive characteristics for vast range of applications, these characteristics such as its shorter decoding delay and no decoding buffer requirements [21]. Moreover, the encoding and the decoding in the IDNC is done using bitwise binary XOR, which requires simple circuits in both the encoder and the decoder sides. One scheme to find the possible IDNC files combination is the IDNC graph which will be discussed in the next section.

1.2.2 IDNC Graph

In order to model the completion problem in a system that consists of a transmitter (a server in our literature) storing a set of files and a set of receivers (clients in our literature). The sever aims to deliver a subset of the packets (files in our literature) missed by each client. We need to find an illustration of the possible file combinations, those are decodable at the same instant by all the clients or any subset of them. In [21, 32, 33] a graph model was introduced to represent the possible IDNC file combinations called IDNC graph. The IDNC graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is an undirected graph [3] consists of a finite set of vertices $\mathcal{V}$ and adjacency edges $\mathcal{E}$, each vertex v_{jk} represents a wanted file f_k by a given client c_j and each edge

[3] An undirected graph is a graph in which edges have no orientation.

connects two vertices $v_{j_1k_1}$ and $v_{j_2k_2}$ represents an opportunity to perform XORing between file f_{k_1} and file f_{k_2}. The server can built the IDNC graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ each time it needs to select an IDNC file combination. Based on the feedback from the clients, the server will build the *log matrix* $L(s)$ defined as follows:

$$L(s) = [l_{j,k}] = \begin{cases} 0, & \text{client } j \text{ already has file } k \\ 1, & \text{client } j \text{ wants file } k \\ -1, & \text{otherwise} \end{cases} \tag{1.1}$$

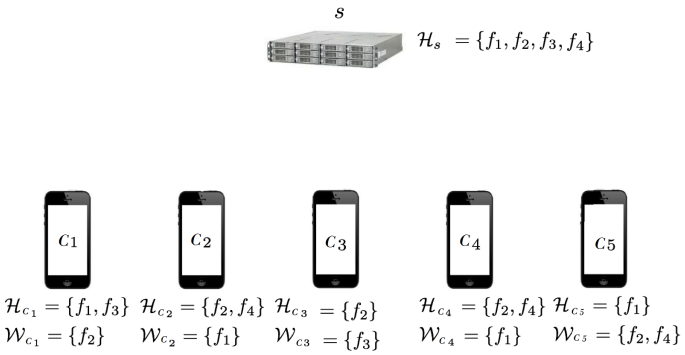
Each wanted file will be represented by a vertex in the IDNC graph, any two vertices v_{j_1,k_1} and v_{j_2,k_2} represent two wanted files f_{k_1} and f_{k_2} induced by to different clients c_{j_1} and c_{j_2} should be set connected by an edge represents encoding opportunity if either one of the following conditions is satisfied:

- C1: file f_{k_1} = file f_{k_2}
- C2: client c_{j_1} already has file f_{k_2} **AND** client c_{j_2} already has file f_{k_1}

Condition (C1) implies that the server can target two different clients c_{j_1} and c_{j_2} by a file f_{k_1} if the two vertices represent the same file, whereas condition (C2) implies that the server can target two different clients c_{j_1} and c_{j_2} by the encoded file $f_{k_1} \oplus f_{k_1}$ and each client will be able to extract its wanted file from it.

The IDNC graph represents all the encoding opportunities and the most preferable transmission pattern is the one induced by XORing all the files corresponding to

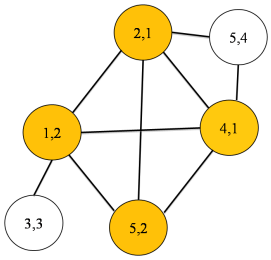
the Maximum Clique [4] in the IDNC graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$.



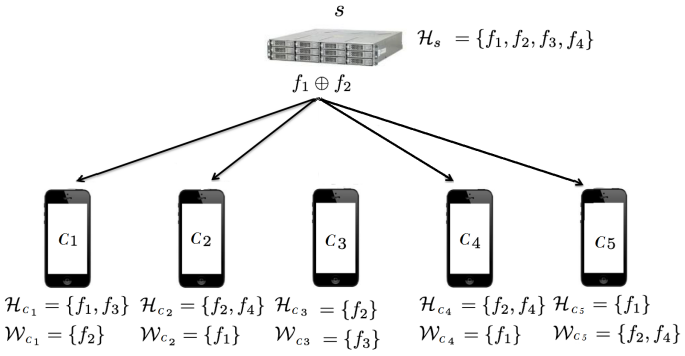
(a) Server content and the clients' requests and contents.

$$L(s) = \begin{array}{c} \begin{matrix} f_1 & f_2 & f_3 & f_4 \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ -1 & 0 & 1 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 1 \end{pmatrix} \end{array} \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{matrix}$$

(b) The coresponding Log matrix.



(c) IDNC graph with maximum clique shown in darker colour



(d) Download pattern corresponding to the maximum clique in the IDNC graph.

Figure 1.2: IDNC Graph Formulation - Numerical Examble.

[4] A clique is a group of vertices in the same graph in which every pair of vertices is adjacent [34].

Fig.1.2 illustrate a numerical example of the IDNC graph formulation. A server stores $\{f_1, f_2, f_3, f_4\}$ and and 5 clients with the following requests and contents:

client c_1 already has $\{f_1, f_3\}$, and wants to download $\{f_2\}$.

client c_2 already has $\{f_2, f_4\}$, and wants to download $\{f_1\}$.

client c_3 already has $\{f_2\}$, and wants to download $\{f_3\}$.

client c_4 already has $\{f_2, f_4\}$, and wants to download $\{f_1\}$.

client c_5 already has $\{f_1\}$, and wants to download $\{f_2, f_4\}$.

Based on these contents and requests the server builds the Log matrix as shown in Fig.1.2b. The IDNC graph for the system at this state shown in Fig.1.2c, the vertices represent the maximum clique are marked by darker colour, the selected vertices induced by files f_1, f_2. Hence, The IDNC combination $f_1 \oplus f_2$ will be sent and some bits will be added to the combination by the server to inform the clients that this combination represents $f_1 \oplus f_2$. The corresponding download schedule is shown in Fig.1.2d and each client will decode the IDNC file as follows :

- Clients c_1 and c_5 already have f_1, so they can extract f_2 by $f_1 \oplus (f_1 \oplus f_2) = f_2$.
- Clients c_2 and c_4 already have f_2, so they can extract f_1 by $f_2 \oplus (f_1 \oplus f_2) = f_1$.
- Client c_3 already has f_2 and the wanted file chunk is f_3, so the combination $f_1 \oplus f_2$ will not serve this client.

The download schedule of the IDNC requires 3 transmission slots to deliver all the required files. In the other hand, the required transmission slots is 4 if no coding tacks place.

1.2.3 IDNC Conflict Graph

IDNC conflict graph is an undirected graph that represents the encoding conflicts. Since any pair of vertices represent two files can be either encoded together or not, the IDNC conflict graph will be the complement of the IDNC graph. The server generates the IDNC conflict graph as a set of vertices each vertex represent a wanted file and any two vertices v_{j_1,k_1} and v_{j_2,k_2} should be set connected by an edge represents encoding conflict if both of the following conditions apply:

- $f_{k_1} \neq f_{k_2}$
- client c_{j_1} does not possess f_{k_2} **OR** client c_{j_2} does not possess f_{k_1}

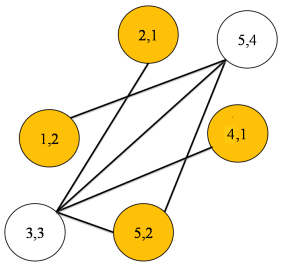


Figure 1.3: IDNC Conflict graph, dark colour representes vertices of the maximum independent set.

These conditions are the complement of the encoding conditions discussed in the previous section. The maximum independent set [5] of the IDNC conflict graph is represented by the same set of vertices which forms the maximum clique in the IDNC opportunities graph. Hence, the download patterns corresponding to the maximum independent set of the IDNC conflict graph and the maximum clique of the IDNC graph are identical. Fig.1.3 shows the IDNC conflict graph for the system in Fig.1.2.

[5] An independent set is a group of pairwise nonadjacent vertices [34].

1.3 Thesis Scope

1.3.1 System Model and Parameters Definition

System and Data Model

The system model of interest comprises a distributed storage network including a set of N_s storage servers $\mathcal{S} = \{s_i\}_{i=1}^{N_s}$, a set of N_f files $\mathcal{F} = \{f_k\}_{k=1}^{N_f}$, and a set of N_c clients $\mathcal{C} = \{c_j\}_{j=1}^{N_c}$. Each server s_i stores a subset of $\mathcal{F}$ named the server Has and it is denoted by $\mathcal{H}_{s_i}$. Any file in $\mathcal{F}$ can be duplicated among the Has sets of the servers. For this model, it is assumed that all the files in the library are stored in the servers collectively (i.e., $\bigcup_{i=1}^{N_s} \mathcal{H}_{s_i} = \mathcal{F}$). Each client c_j also stores some files in its Has set ($\mathcal{H}_{c_j}$) from previous downloads. For simplicity of analysis, it is assumed that files are equally sized, and that all servers have the same storage capacity. The file download time is measured in download time-units (DTU). The storage servers (transmitters) are assumed to operate on orthogonal channels and at each download time epoch, the reception frequency of each client is tuned to only one server.

Parameters and Definitions

Let p_{ij} be the file corruption probability observed by client c_j on the down-link channel with server s_i. Also, let q_{ij} be the feedback corruption probability observed by client c_j on the up-link channel with server s_i. It is intuitive to assume that $q_{ij} < p_{ij}$ due to the larger size of the files on the down-link as compared

to up-link, and the lower data rates at the up-link. Let P_s be the probability that a given file is stored at a given server. Since all the servers have the same storage capacity and each server stores ($S = |\mathcal{H}_{s_i}|$) of non-duplicated files, P_s can be expressed as

$$P_s = \frac{S}{N_f}. \tag{1.2}$$

Let $H_j = |\mathcal{H}_{c_j}|$ be the side information at the client c_j. The average side information at all clients can be expressed as $H = (\frac{1}{N_c}) \sum_{j=1}^{N_c} H_j$ (the average number of the files already stored at the client), then the probability that a given file is already stored at a given client (P_c) can be expressed as

$$P_c = \frac{H}{N_f}. \tag{1.3}$$

In addition, let $D_j = |\mathcal{W}_{c_j}|$ be the number of the demanded (wanted) files by client c_j. The average number of the wanted files by all clients can be expressed as $D = (\frac{1}{N_c}) \sum_{j=1}^{N_c} D_j$ (the average number of the files wanted and not downloaded by the client), then the probability that a given file is wanted by a given client (P_w) can be expressed as

$$P_w = \frac{D}{N_f}. \tag{1.4}$$

Finally, the repetition index $1 \leq R \leq N_s$ can be defined as the average number of file copies stored in all servers, and can be expressed as

$$R = \frac{N_s S}{N_f} = N_s P_s. \tag{1.5}$$

1.3.2 problem statement

In this section we will discuss the main motivations that lead to this work and how they are important to the field of research in network coding area. As can be noticed from the literature, all the previous studies have considered IDNC network coding scheme to minimize the completion delay in point-to-multipoint systems. Despite the merits of these studies, they did not consider multipoint-to-multipoint systems model. The conventional IDNC has shown remarkable improvements in the performance of point-to-multipoint networks. However, applying the conventional IDNC on multipoint-to-multipoint system may caused simultaneous transmissions to the same client. Since the client can decode only one IDNC file at each DTU, these transmissions are resources squandering, and they lead to performance and efficiency degradation. The next section shows a numerical example to show the necessity of conflict-free transmission in any multipoint-to-multipoint system adopts IDNC network coding.

1.3.3 Motivating Example

Let us consider the simple example shown in Fig. 1.4, which includes 3 storage servers having $\mathcal{H}_{s_1} = \{f_1, f_2, f_5\}$, $\mathcal{H}_{s_2} = \{f_1, f_4, f_5\}$, $\mathcal{H}_{s_3} = \{f_2, f_3, f_5\}$, and 6 clients having the following contents and requests:

$\mathcal{H}_{c_1} = \{f_1\}$, and $\mathcal{W}_{c_1} = \{f_5\}$.

$\mathcal{H}_{c_2} = \{f_1, f_5\}$, and $\mathcal{W}_{c_2} = \{f_4\}$.

$\mathcal{H}_{c_3} = \{f_3, f_5\}$, and $\mathcal{W}_{c_3} = \{f_2\}$.

$\mathcal{H}_{c_4} = \{f_4\}$, and $\mathcal{W}_{c_4} = \{f_1\}$.

$\mathcal{H}_{c_5} = \{f_1, f_3\}$, and $\mathcal{W}_{c_5} = \{f_2\}$.

$\mathcal{H}_{c_6} = \{f_1, f_2\}$, and $\mathcal{W}_{c_6} = \{f_3\}$.

Assuming no corruption occurs, without using network coding at this system, the download process requires 2 DTUs and can be performed as follows:

- DTU 1 :
  - s_1 sends f_2: to c_3 AND c_5.
  - s_2 sends f_1: to c_4.
  - s_3 sends f_3: to c_6.
- DTU 2 :
  - s_1 sends f_5: to c_1.
  - s_2 sends f_4: to c_2.

In the conventional PMP network coding case (as in [20]), based on the feedback from the clients, each server s_i will generate its *log matrix* $L(s_i)$ as follows

$$L(s_i) = [l_{j,k}] \; \forall \; c_j \in \mathcal{C} \text{ and } f_k \in \mathcal{H}_{s_i} \quad s.t. \quad l_{j,k} = \begin{cases} 0, & c_j \text{ already has } f_k \\ 1, & c_j \text{ wants } f_k \\ -1, & \text{otherwise} \end{cases} \tag{1.6}$$

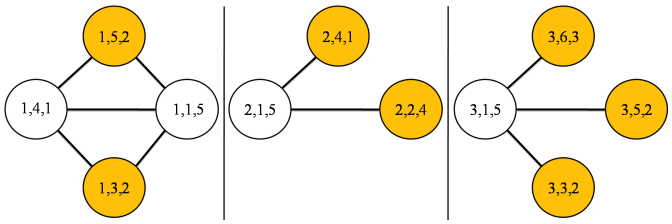
Given the log matrices, each server can build its IDNC conflict graph given its own log matrix.

Any two vertices v_{j_1,k_1} and v_{j_2,k_2} are set connected by an edge represents coding conflict if both of the two following conditions are satisfied:

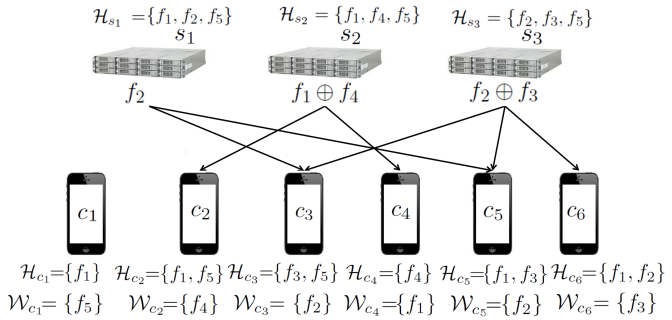
- $f_{k_1} \neq f_{k_2}$
- $f_{k_1} \notin \mathcal{H}_{c_{j_2}}$ OR $f_{k_2} \notin \mathcal{H}_{c_{j_1}}$

Note that these conditions represent logical complementary conditions to the ones used in generating the conventional IDNC graph in [20].

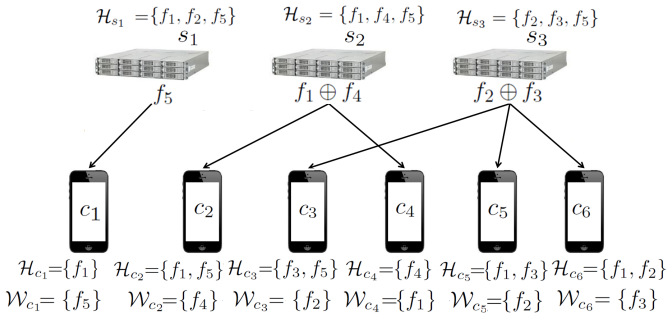
Each server finds its maximum independent set and transmits the corresponding coded files to the targeted clients. Fig. 1.4a illustrates the separate IDNC conflict graphs built by the servers and the resultant maximum independent sets (shown in darker colour in the figures).



(a) Separated IDNC conflict graphs of s_1, s_2 and s_3 respectively.



(b) Download pattern corresponding to separated IDNC solution.



(c) Conflict-free transmission pattern to serve the clients.

Figure 1.4: Motivating Example

In this case, the servers transmitting pattern in the first DTU will be:

- s_1 sends f_2: to c_3 AND c_5.
- s_2 sends $f_1 \oplus f_4$: c_2 AND c_4 can decode it.
- s_3 sends $f_2 \oplus f_3$: c_3, c_5 AND c_6 can decode it.

This download pattern is shown in Fig. 1.4b. The figure depicts several cases where the same client are served by two different servers. For instance, servers s_1 and s_3 both delivers f_2 to client c_3 which can only receive from one of them. This event will be called a *transmission conflict* which is undesirable. At the same time, c_1 is not served by non of the servers, this will make the overall download time equals to 2 DTUs. As a result, the no coding scenario and the conventional IDNC have the same performance.

This example raises the question of whether a better download pattern can be found to reduce the download time. By careful examination, the files download process can be achieved in only 1 DTU if the servers adopt the following download pattern and coded transmissions:

- s_1 sends f_5: to c_1.
- s_2 sends $f_1 \oplus f_4$: c_2 AND c_4 can decode it.
- s_3 sends $f_2 \oplus f_3$: c_3, c_5 AND c_6 can decode it.

This transmission pattern is shown in Fig. 1.4c, where it can be noticed that only 1 DTU is needed to serve all clients. Thus, a better solution can be obtained when transmission conflicts are compromised, despite that may result in a sub-optimal coding schedule for some of the servers. The question is how to find such a solution for an arbitrary number of clients, files and servers. A novel dual-conflict graph model that will be introduced in the next chapter can be used to find these download pattern and coded transmissions systematically.

1.4 Thesis Contributions

To the best of our knowledge, this thesis introduces the first conflict-free IDNC algorithm suitable for distributed storage systems (or any multipoint-to-multipoint system) involving transmissions from multiple transmitters. Our contributions on this work can be summarized into three points as follows:

- We design a novel conflict-free IDNC suitable for file download from any MPM system. The completion delay problem is formulated as a SSP problem and for simplicity, a heuristic channel-aware algorithm is designed to solve the completion delay problem in such systems. The simulation results show that the proposed conflict-free IDNC achieves a remarkable enhancement in the average download time required to deliver the missed files, compared to the application of the conventional IDNC approach separately at the servers. This work was published in [35].

- We extend the system model to involve lossy feedback environment. The SSP problem formulation in perfect feedback is updated to a POSSP formulation reflecting the uncertainties resulting from unheard feedback events in the lossy feedback environment. A maximum Likelihood estimator is derived to minimize the completion delay under LF environments.

- We approximate the chromatic number of the new dual conflict graph. This enable as to find meaningful lower and upper bounds of the performance of the proposed conflict-free IDNC algorithm.

1.5 Thesis Organization

The rest of the thesis is organized as follows. The second chapter introduces a novel conflict-free IDNC algorithm based on a graph model that guarantees conflict-free IDNC transmissions, the completion delay problem in Multipoint-to-Multipoint systems is formulated as a SSP problem and for solution simplicity the chapter introduces a proposed heuristic algorithm to solve it. The completion delay problem in a Multipoint-to-Multipoint system with Lossy Feedback (LF) environments is studied in chapter 3. The completion delay minimizing policy based on Maximum Likelihood rule in such environment is derived and also two other possible minimizing policies are studied in the simulations. In chapter 4, the chromatic number of the dual conflict graph and lower and upper bounds of the performance of the proposed algorithm are derived. Finally, chapter five conclude the work by highlighting the main contributions and conclusions acquired in this thesis. It also suggests some future works in the field of network coding.

CHAPTER 2

CONFLICT-FREE IDNC NETWORK CODING

2.1 Motivation

This chapter introduces a systematic approach that enables the transmitted server to select conflict-free IDNC files combination in a distributed storage network or any multipoint-to-multipoint system. A heuristic channel-aware algorithm is designed to solve the completion delay problem based on maximum weighted vertex search scheme. The proposed algorithm can be implemented in centralized and distributed forms. This can be achieved with a significant reduction in the average download time compared to using the conventional IDNC approach at each server separately.

2.2 Conflict-Free IDNC Network Coding

In order to avoid transmission conflicts, we need to find a comprehensive representation that includes transmission and coding conflicts in one paradigm. The *dual conflict IDNC graph* model is proposed to represents these two types of conflicts in one augmented graph model. The proposed dual conflict IDNC graph $\mathcal{G}$ consists of a set of vertices and a set of undirected edges generated as follows:

Vertex Set:

Given the augmented log matrix $L(s_1, s_2, \ldots, s_{N_s})$ (constructed by aligning the log matrices of all the servers, shown in Fig. 2.1a for the example in Fig. 1.4), create a vertex $v_{i,j,k}$ $\forall$ $c_j \in \mathcal{C}$ and $f_k \in \mathcal{H}_{s_i} \cap \mathcal{W}_{c_j}$ (Only files that are required by c_j and at the same time, it is stored in the server s_i.).

Coding Conflict Edges:

Any pair of vertices v_{i,j_1,k_1} and v_{i,j_2,k_2} in $\mathcal{G}$, are set to be adjacent by an undirected edge that represents a coding conflict both of them are induced by the same server and the following condition applies:

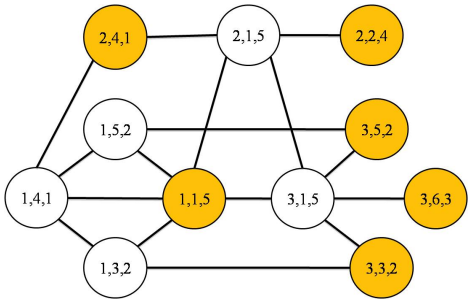
- $f_{k_1} \neq f_{k_2}$ but $f_{k_1} \notin \mathcal{H}_{c_{j_2}}$ OR $f_{k_2} \notin \mathcal{H}_{c_{j_1}}$. In this case, f_{k_1} and f_{k_2} are wanted by two different clients. However, one client at least does not store the other wanted file. Accordingly, the combination $f_{k_1} \oplus f_{k_2}$ will not be useful for the client to extract its own requested file.

Transmission Conflict Edges:

Any pair of vertices v_{i_1,j_1,k_1} and v_{i_2,j_2,k_2} in $\mathcal{G}$, are set adjacent by an undirected edge that represents a transmission conflict if $j_1 = j_2$ AND $i_1 \neq i_2$. In other words, any pair of vertices symbolizing the simultaneous transmission from two different storage servers to the same client will be set adjacent.

$$L(s_1,s_2,s_3) = \begin{array}{c} \overbrace{}^{s_1}\ \overbrace{}^{s_2}\ \overbrace{}^{s_3} \\ \begin{matrix} f_1 & f_2 & f_5 & f_1 & f_4 & f_5 & f_2 & f_3 & f_5 \end{matrix} \\ \begin{bmatrix} 0 & -1 & 1 & 0 & -1 & 1 & -1 & -1 & 1 \\ 0 & -1 & 0 & 0 & 1 & 0 & -1 & -1 & 0 \\ -1 & 1 & 0 & -1 & -1 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & -1 & -1 & -1 & -1 \\ 0 & 1 & -1 & 0 & -1 & -1 & 1 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1 & -1 & 0 & 1 & -1 \end{bmatrix} \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{matrix} \end{array}$$

(a) The augmented log matrix of s_1, s_2 and s_3.



(b) Dual-conflict IDNC graph of the example in Sec. 1.3.3. dark colour represents vertices of the maximum independent set.

Figure 2.1: The augmented matrix and the proposed dual conflict graph for the example in Sec. 1.3.3.

Any maximal independent set in this graph will be without both transmission and coding conflicts. Moreover, The most preferable transmission schedule is the one induced by the maximum independent set in $\mathcal{G}$. Fig. 2.1b shows the dual-conflict graph of the example in Sec. 1.3.3, and the maximum independent set (represented by darker colour in the figures). In this case the download pattern will be identical to that in Fig. 1.4c.

2.3 Proposed Solution using Dual-Conflict Graph

Let us define the system state in which all the clients have received all the wanted files as the *goal-state* $\mathcal{S}^*$. Assuming that at any arbitrary time instant t the system operates at non-goal state $\mathcal{S}^t$ (the remoteness of $\mathcal{S}^t$ from the goal-state $\mathcal{S}^*$ is subject to the want set of each client). Our goal is to fetch the system as fast as possible to the goal-state $\mathcal{S}^*$, with minimum number of transitions (DTUs). The system moves from a given state to another state based on the transmitted IDNC file, to this end the problem is a shortest path problem. However, the erasure events occur in the forward sub-channel between the transmitted server and the targeted client impose randomness on the transition to another state or even staying at the same state (if all the client did not receive the IDNC file). Consequently, the minimum completion delay problem in the perfect feedback environment can be formulated as a SSP problem as shown in the next section.

2.3.1 SSP Formulation for Perfect Feedback Environment

SSP is a class of probabilistic planning problems which describe a wide range of possible scenarios where the purpose of the active agent is to reach a goal state in the minimum costly way from any non-goal state using actions with probabilistic outcomes. The minimum completion delay problem has the same problem structure as a SSP problem where the possible state space, action space, transition probabilities and the state-action cost can be defined as follows.

- *State Space*:

  Due to the fact that the client can tune to a single server at each time epoch, states space can be considered as all the possibilities of the augmented log matrix that could take place until completion.

- *Action Space*:

  The action space for each state can be defined as the set that contains all possible dual-conflict IDNC graph's maximal independent sets can be generated from the augmented log matrix of this state.

- *State-Action Transition Probabilities*:

  The system will stay at the same state or transit to another state based on the action (i.e., transmission pattern corresponding to selecting a given IDNC file) taken at any state. This action identifies the targeted clients by the servers, consequently it identifies the sub-channels and and the probability of files corruption between each client and the serving server.

- *State-Action Cost*:

  One DTU is the cost of each action taken by the servers towards the goal-state $\mathcal{S}^*$.

It is clear that this SSP has the dimensionality problem due to the size exponential growth of both action space $O\left(3^{\frac{N_s N_c N_f}{3}}\right)$ and the state space $O\left(2^{N_c N_f}\right)$. Accordingly, solving this problem optimally is intractable, and an efficient heuristic algorithm is required to solve it.

2.3.2 Maximum Weighted Vertex Search Algorithm

Based on the fact that the process of finding the maximum independent set in a given graph is considered as NP-hard problem, a heuristic algorithm that finds a maximal independent set through a maximum weight vertex search approach is designed. From IDNC's properties, the download time is lower bounded by the want set size of the client that needs the greatest number of files. Consequently, we need to address the maximum number of clients with large want sets in each transmission. Furthermore, in lossy environment, the client needs to tune to the server with the best channel. We first assign a raw weight w'_{ijk} to each vertex $v_{i,j,k}$ in $\mathcal{G}$. This weight reflects the size of the wants set of c_j, and also the probability of file reception at client c_j from server s_i. This raw weight is defined as follows

$$w'_{ijk} = (1 - p_{ij})|\mathcal{W}_j|. \tag{2.1}$$

Where, the raw weight defined in (2.1) can be used for both cases, one or multiple file request (in the former case, $|\mathcal{W}_j| = 1$), and it also applicable for both corruption-free and corrupted environments (in the former environment, $p_{ij} = 0$). We define the non-adjacency indicator of vertices v_{i_1,j_1,k_1} and v_{i_2,j_2,k_2} ($a_{i_1j_1k_1,i_2j_2k_2}$) as follows

$$a_{i_1j_1k_1,i_2j_2k_2} = \begin{cases} 1, & v_{i_1,j_1,k_1} \text{ is not connected to } v_{i_2,j_2,k_2} \\ 0, & v_{i_1,j_1,k_1} \text{ is connected to } v_{i_2,j_2,k_2}. \end{cases} \tag{2.2}$$

Finally, the vertex modified weight w_{ijk} is defined as

$$w_{ijk}(\mathcal{G}) = w'_{ijk} \sum_{\forall v_{xyz} \in \mathcal{G}} w'_{xyz} \cdot a_{ijk,xyz}. \tag{2.3}$$

Hence, a vertex with the highest modified weight has two attractive aspects. The first aspect, it is disjointed to a large number of vertices. The second aspect, these vertices are induced by clients with high raw weight w'_{ijk}. With these definition

Algorithm 2.1 Maximum Weighted Vertex Search Algorithm

1: Initialization:
- Set Graph $\mathcal{G}_s \leftarrow \mathcal{G}$.
- $\forall$ vertices in $\mathcal{G}$, compute the raw weights as in (2.1).
- Set the Selected Independent set $\Gamma = \phi$.

2: **repeat**
3: $\forall$ vertex $v_{ijk} \in \mathcal{G}_s$: Calculate $w_{ijk}(\mathcal{G}_s)$ as in (2.3)
4: $v^*_{ijk} \leftarrow \max_{v_{ijk} \in \mathcal{G}_s}\{w_{ijk}(\mathcal{G}_s)\}$
5: $\Gamma = \Gamma \cup v^*_{ijk}$
6: $\mathcal{G}_s \leftarrow \mathcal{G}_s \setminus \left(v^*_{ijk} \cup \mathcal{N}_{\mathcal{G}_s}(v^*_{ijk})\right)$
7: **until** $\mathcal{G}_s = \phi$, *end repeat.*
8: *Return the Selected independent set* Γ

of the modified weights, the algorithm in Algorithm 2.1 executes iteratively. Each iteration will be implemented as follows: first, compute the modified weight to each vertex in the graph. Second, the maximum weighted vertex v^*_{ijk} with the maximum modified weight is selected and added to Γ. Finally, the new subgraph $\mathcal{G}_s$ is formed by eliminating the chosen vertex v^*_{ijk} and all the vertices those are connected to v^*_{ijk} (symbolized in the algorithm by $\mathcal{N}_{\mathcal{G}}(v_{ijk})$) from the old subgraph. The algorithm continues running until there is no more vertices in the subgraph $\mathcal{G}_s$.

2.3.3 Implementation Issues

We can implement the proposed conflict-free IDNC algorithm in either centralized or decentralized (distributed) scenario. In the former scenario, a cloud-storage controller (that knows the side information of all clients and the stored content at the servers) can build the dual-conflict graph and use Algorithm 2.1 to schedule the conflict-free IDNC combinations at each DTU. In the latter scenario, each server can build the dual-conflict graph separately, by exchange their knowledge of content and the clients sub-channels parameters with each other using their backbone connections. Each server needs to run Algorithm 2.1 separately and finds the conflict-free IDNC combinations that are required to be transmitted at each DTU. Since the update rate of distributed storage networks content is significantly low compared to the file download time DTU, our decentralized scenario can be maintained and work appropriately at each server. In perfect feedback environment, the download patterns resulted from centralized and distributed scenarios of the conflict-free IDNC algorithm are identical. However, in lossy feedback environment, the uncertainties induced by the unheard feedback lead to variety in the download patterns. This is resulted from the variety of the decision in the centralized and the distributed implementation scenarios, as will be shown in Sec. 4.4.

2.4 Simulation Results

In this section, we introduce the simulation results of the dual-conflict IDNC algorithm and the conventional PMP IDNC solution both applied to a distributed storage system with $N_s = 10$ servers. In each simulated scenario, each client wants to download a random number of files, the average value of the wanted files equals to $\lceil \frac{N_f}{3} \rceil$ files. At the same time, each client already has a random number of files, the average number of the downloaded files equals to $\lceil \frac{N_f}{3} \rceil$ files.

We assume that the probability of file corruption at the downlink channel between any arbitrary server and client is uniformly distributed and lies in the interval $[0.01, 0.3]$. During the complete file(s) download time, this file corruption probability remains unchanged. The uplink channel between any arbitrary client and server is assumed to be corruption-free.

Fig. 2.2 illustrates the average completion delay versus the number of clients N_c, with $N_f = 100$ files and each server stores 50 files which are taken randomly from the files library that consists of 100 files. The figure depicts a remarkable reduction in the average completion delay when the conflict-free algorithm is compared with the conventional PMP scheme, a reduction of 7 to 10 DTUs can be noticed. We can also observe that the achieved performance degrades as the total number of clients increases. This can be explained by the fact that with increasing the number of clients with the same number of files in each server, the transmission conflicts will be fewer when the conventional IDNC approach is utilized. This enhances its overall performance and makes it slightly closer to that of the

conflict-free scheme. Fig. 2.3 depicts a similar comparison over a range of the

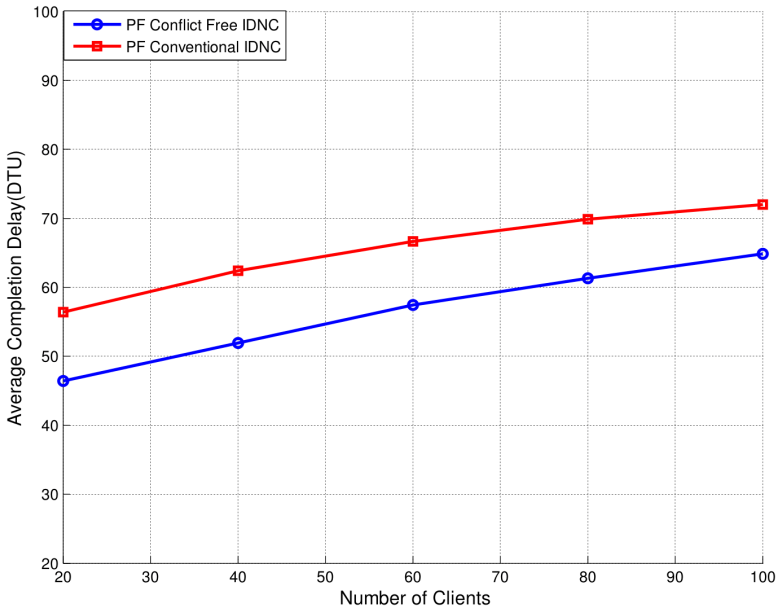


Figure 2.2: The average completion delay versus the number of clients N_c.

total number of files N_f, with $N_c = 60$ clients and each server stores $\frac{N_f}{2}$ files. We can notice performance gains achieved by the conflict-free IDNC approach in terms of reducing the average download time for all tested values of N_f.

Fig. 2.4 illustrates the effect of the storage capacity of the server on the performance of both the conventional and the conflict-free IDNC approaches with fixed library size of $N_f = 100$ files and $N_c = 60$ clients. It can be observed from the figure that the performance of the conventional IDNC approach increases as the server size increases, while the conflict-free IDNC approach performance is enhanced. This can be illustrated by the fact that the server storage capacity increment allows the server to target more clients in its selected IDNC combinations in every DTU. However, when each server finds its IDNC combinations individually, with the increase in the number of files in each server, it becomes more likely

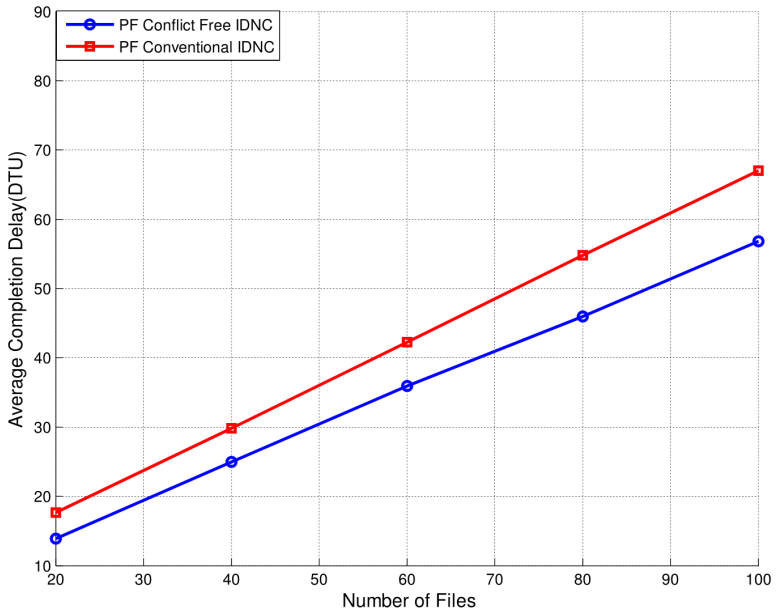


Figure 2.3: The average completion delay versus the number of files N_f.

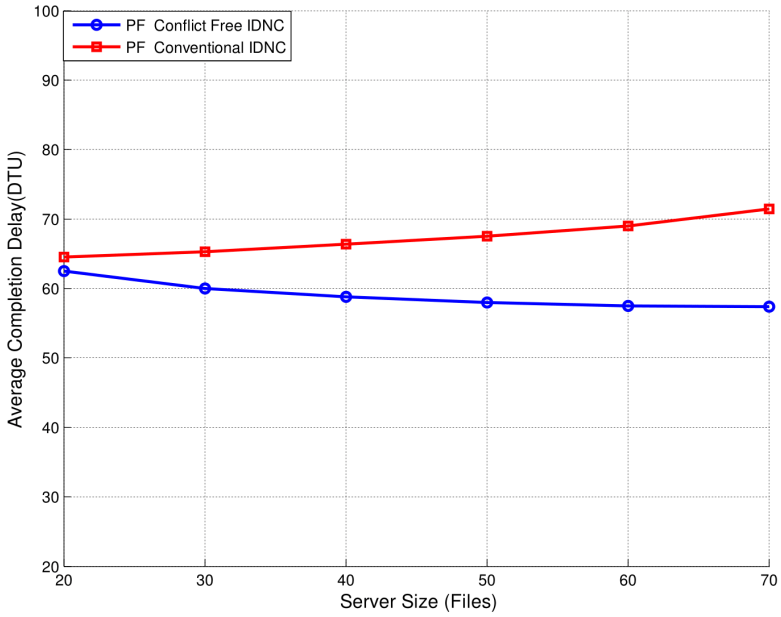


Figure 2.4: The average completion delay versus the server size.

that two servers would serve the same client (i.e., a transmission conflict would occur). So a smaller number of clients will be served on average and more time is required to serve all the clients.

2.5 Summary and Conclusions

In this chapter we introduced a novel graph model that represents both the transmission conflicts and the coding conflicts in one paradigm that is called the dual-conflict graph. We can generate conflict-free IDNC coded transmission scheduling that can provide more efficient scheduling scheme to serve the clients. These conflict-free transmissions are generate by finding the maximum independent sets in the proposed graph model. To find the maximum independent set in this graph, a heuristic channel-aware algorithm is proposed to solve the download time minimization problem after showing that its optimal solution is intractable. Simulation results illustrated that our proposed conflict-free IDNC algorithm is superior to the conventional IDNC in reducing the overall download time.

The next chapter introduces asymptotic upper and lower bounds of the performance of the conflict-free IDNC algorithm. This is done with the assumption that each client wants to download only one file. A comparison between the performance of the proposed solution in 2.3.2 and the optimum solution that was founded using BronKerbosch algorithm is also introduced.

CHAPTER 3

ASYMPTOTIC UPPER AND LOWER BOUNDS OF THE PERFORMANCE OF THE CONFLICT-FREE IDNC ALGORITHM

3.1 Motivation

In this chapter, asymptotic lower and upper bounds of the conflict-free IDNC algorithm is derived. This was done based on the chromatic number of the dual conflict graph for a certain fixed placement policy of files in the servers. We find an approximation to the chromatic number of the dual conflict graph (In case

each client wants only one file) using random graph theory. The performance of the conflict-free IDNC algorithm using the heuristic algorithm investigated in Sec. 2.3.2 also compared with the optimum solution of the conflict-free IDNC algorithm. We use Bron-Kerbosch algorithm to find the maximum independent set in the dual conflict graph which leads to the optimum solution of the conflict-free IDNC algorithm.

The reminder of this chapter is organized as follows: Section 3.2 introduces a brief introduction to the random graph theory and random graph models. In Section 3.3 the definitions of the Binomial distribution, hypergeometric distribution and the multivariate hypergeometric distribution are summarized. Graph coloring and the chromatic number definition are introduced in Section 3.4. In Section 3.5, the chromatic number of the dual conflict graph is derived using random graph theory model. Bron-Kerbosch algorithm summarized in Section 3.6. Section 3.7 illustrates the simulation results that show that the bounds are valid for both the fixed file placement and the random file placement policies in the servers. Finally, Section 3.8 concludes the chapter by highlighting the main contributions and results.

3.2 Random Graphs

Random graphs theory initiated in [36], the authors addressed the question of the probability of a random graph being connected. This key question is not only interesting from a mathematical perspective, it triggered a new evolution of graphs modeling to provide an exceptionally quantifiable graph model. Random graph model provides convenient approximations to the average values of the graph characteristics [37]. This enable us to avoid the complexity induced by conventional constrained graph model. A random graph $\mathcal{G}_{\nu,M}$ can be defined as a graph that is formulated by starting with a finite set of isolated vertices ν and adding M successive edges between these vertices randomly [36]. Another random graph model was introduced by [38]. In this model, a random graph can be defined as a finite set of vertices and edges in which every possible edge occurs independently with probability $0 \leq \pi \leq 1$. Each possible edge between any pair of vertices will occur with probability π and disappear with probability $1-\pi$. For fixed π, the two models are almost interchangeable since the maximum number of possible edges in a given graph with ν vertices is given by $\binom{\nu}{2}$. The number of the graph's edges can be approximated as $M = \pi\binom{\nu}{2}$ and the graph in the former model can be defined as $\mathcal{G}_{\nu,\pi\binom{\nu}{2}}$. Random graph models are useful to extract the properties of typical graph. These models enable the researchers to extract the graphs properties without the intractability induced by the traditional schemes. Moreover, random graph is considered as one of the best models used in modeling the real-world networks such as the social networks and the Internet [39].

3.3 Probability Theory Background

The random graph theory lies at the intersection between the probability theory and the graph theory. In this section, some definitions of probability distributions are recalled. These distributions are involved in the derivation of the total connectivity probability in the dual conflict graph.

3.3.1 Binomial Distribution

A random variable X is said to follow binomial distribution if it represents the number of the possible successes in a finite sequence of n independent true or false experiments. In every experiment, the success event occurs with fixed probability p and the failure event occurs with probability $1-p$. The binomial random variable X is symbolized by X $\sim$ $Bin\,(n,p)$ and the probability of winning precisely k successes in n independent experiments can be described as :

$$\mathbb{P}\,(\mathbf{x}=k)=\binom{n}{k}p^{k}\,(1-p)^{n-k}\,. \tag{3.1}$$

A special case of the true or false experiment with only one trail (when n = 1) is a Bernoulli trial or Bernoulli experiment [40].

3.3.2 Hypergeometric Distribution

In probability theory, the discrete random random variable is said to be hypergeometric if it characterizes the probability of getting k successes out of n samples and at each sample the success event has different probability. The sampling is carried out in a finite population of size N that includes exactly K successes, wherein each sample is either a success or a failure. The following conditions characterize the hypergeometric distribution:

- The outcome of each sample can be categorized into one of two mutually exclusive events (e.g. True-False or Pass-Fail or Employed-Unemployed Female/Male or Success-Failed).
- The probability of a success varies on each sample, i.e., as each sample diminishes the population size (such as sampling without replacement from a finite population).

A random variable X that satisfies the aforementioned characteristics is called hypergeometric random variable and its probability density function (pdf) can be expressed as:

$$\mathbb{P}\left(\mathbf{x} = k\right) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}. \tag{3.2}$$

where,

- N: represents the size of the population,
- K: represents the total number of success states included in the population,

- n: represents the number of carried out samples,
- k: represents the number of observed successes out of n samples.

The hypergeometric distribution is used in modeling the number of possible successes events in a sample space of size n that are drawn without replacement. The draws are not independent from a population of finite size. On the other hand, sampling with replacement results in independent draws and so the resulting random variable will follow binomial distribution, not a hypergeometric one [40].

3.3.3 Multivariate Hypergeometric Distribution

The Multivariate Hypergeometric distribution is considered as an extension of the hypergeomeric distribution at which more than two different states or sub-group of individuals in a group are existed. Let us assume a multivariate hypergeometric process consist of $d_1, d_2, d_3, \cdots, d_k$ different individual types or subgroups in a population D. The probability of getting exactly x_1 out of d_1 subgroup and x_2 out of d_2 subgroup (and so on) in X draws without replacement can be expressed as:

$$\mathbb{P}\left(d_1 = x_1, d_2 = x_2, \cdots, d_k = x_k\right) = \frac{\prod_{i=1}^{k} \binom{d_i}{x_i}}{\binom{D}{X}}, \tag{3.3}$$

where

$$D = \sum_{i=1}^{k} d_i, \qquad X = \sum_{i=1}^{k} x_i.$$

3.4 Graph Coloring and The Chromatic Number

In graph theory, graph coloring is one type of graph labeling; each element of a graph is assigned a label traditionally called color based on certain conditions. One simple form of graph coloring is vertex coloring. In this form, each vertix in the graph is assigned a color subject to the constraint that adjacent vertices do not share the same color. The chromatic number of graph $\mathcal{G}$ is denoted by $\chi(\mathcal{G})$ and can be defined as the smallest number of labels or colors required to color the vertices of graph $\mathcal{G}$ such that any two adjacent vertices do not share the same color. In our model, the chromatic number of the dual-conflict IDNC graph represents the number of the possible conflict free download patterns. These download patterns include the optimum download pattern induced by the maximum independent set in the dual-conflict IDNC graph. In a system without file repetition within the servers ($R = 1$) and each client wants more than one file, the chromatic number is an upper bound of the conflict free IDNC algorithm performance. That is because each transmitted IDNC file will generate new encoding opportunities, while the chromatic number represents the number of the transmissions without considering the new generated encoding opportunities.

3.5 The Chromatic Number of the Dual Conflict Graph

In this section, dual-conflict graph chromatic number is derived using random graph theory model. In [37], it was proven that every random graph $\mathcal{G}_{\nu,\pi}$, with a given number of vertices ν and a fixed connectivity probability π (π is the probability that any pair of vertices is connected), has a chromatic number $\chi\left(\mathcal{G}_{\nu,\pi}\right)$ that can be approximated as

$$\chi\left(\mathcal{G}_{\nu,\pi}\right) = \left(\frac{1}{2} + o(1)\right)\ \log\ \left(\frac{1}{1-\pi}\right)\frac{\nu}{\log\ \nu}. \tag{3.4}$$

Theorem 3.1 *The probability π of getting pair of adjacent vertices connected in the dual-conflict graph, with fixed file placement policy in the servers and the case of one file request per client, can be expressed as*

$$\pi = \left(\frac{N_c - 1}{N_c N_s - 1}\right)\left(1 - \frac{N_c - 1}{N_c S - 1}\right)(1-\psi) + \frac{R\left(R-1\right)}{\nu\left(\nu-1\right)}. \tag{3.5}$$

where,

$$\nu = N_c R;$$

$$\psi = \sum_{y=0}^{H} \frac{\binom{H}{y}\binom{N_f - H}{H-y}}{\binom{N_f}{H}} \sum_{e_1 = max\left(0, H-\left(N_f - S\right)\right)}^{min(H-\mathbf{y}, S)} \binom{S}{e_1}\binom{N_f - S}{H-y-e_1}\binom{N_f}{H-y}^{-1}\frac{e_1}{S}\times$$

$$\sum_{e_2 = max\left(0, H-\left(N_f-(S-e_1)\right)\right)}^{min(H-\mathbf{y}, (S-e_1))} \binom{S-e_1}{e_2}\binom{N_f-(S-e_1)}{H-y-e_1}\binom{N_f}{H-y}^{-1}\frac{e_2}{S-1}.$$

Proof: Based on the log matrix $L(s_i)$ of the server s_i, we create a vertex $v_{i,j,k}$ $\forall\ c_j \in \mathcal{C}$ and $f_k \in \mathcal{H}_{s_i} \cap \mathcal{W}_{c_j}$. Thus, the total number of vertices ν in the dual conflict graph in the case of one requested file per client is written as

$$\nu = N_c R. \tag{3.6}$$

To derive π in (3.4), we need to define the conditions needed for two vertices to be connected in the dual conflict graph as events:

- E1: Any two vertices v_{i,j_1,k_1} and v_{i,j_2,k_2}, both of which are induced by the same server, will be set adjacent by an undirected edge represents a coding conflict if
  - C1: $f_{k_1} \neq f_{k_2}$.
  - C2: $f_{k_1} \notin \mathcal{H}_{c_{j_2}}$ OR $f_{k_2} \notin \mathcal{H}_{c_{j_1}}$.
- E2: Any two vertices v_{i_1,j_1,k_1} and v_{i_2,j_2,k_2} will be set adjacent by an undirected edge represents transmission conflict if $c_{j_1} = c_{j_2}$ AND $s_{i_1} \neq s_{i_2}$.

Any two vertices will be set adjacent if E1 is satisfied or if E2 is satisfied. It is clear that E1 and E2 are mutually exclusive events (E1 implies $s_{i_1} = s_{i_2}$ while E2 implies $s_{i_1} \neq s_{i_2}$), so the connectivity probability in the dual conflict graph is expressed as

$$\pi = \mathbb{P}(\text{E1}) + \mathbb{P}(\text{E2}) \tag{3.7}$$

The probability that E1 occurs can be written as

$$\mathbb{P}\left(\mathrm{E1}\right) = \mathbb{P}\left(1s \cap \mathrm{C1} \cap \mathrm{C2}\right) = \mathbb{P}\left(1s\right)\mathbb{P}\left(\mathrm{C1}|1s\right)\mathbb{P}\left(\mathrm{C2}|1s, \mathrm{C1}\right). \tag{3.8}$$

where $1s$ is the event that the two vertices are induced by the same server. To find $\mathbb{P}\left(1s\right)$, let X_i be the number of vertices induced by server s_i. Since the files are distributed among the servers with fixed placement R will be integer and the total files will be stored in a block of servers, the number of blocks in the system is $B = \frac{N_s}{R}$. Consequently, X_i can be modeled as a binomial random variable $Bin\left(N_c, \frac{1}{B}\right)$. The total number of the vertices in $\mathcal{G}$ can be expressed as $\nu = \sum_{i=1}^{N_s} X_i$, and without loss of generality, we can consider that $X_{N_s} = \nu - \sum_{i=1}^{N_s-1} X_i$. Based on these facts, the probability $\mathbb{P}\left(\mathbf{x} = \mathbf{x}'|\sum_{i=1}^{N_s} X_i = \nu\right)$ can be found as follows

$$\begin{aligned}
\mathbb{P}\left(\mathbf{x} = \mathbf{x}'|\sum_{i=1}^{N_s} X_i = \nu\right) &= \frac{\mathbb{P}\left(\mathbf{x} = \mathbf{x}', \sum_{i=1}^{N_s} X_i = \nu\right)}{\mathbb{P}\left(\sum_{i=1}^{N_s} X_i = \nu\right)} \\
&= \frac{\mathbb{P}\left(X_1 = x_1, ...X_{N_s-1} = x_{N_s-1}, X_{N_s} = \nu - \sum_{i=1}^{N_s-1} X_i\right)}{\mathbb{P}\left(\sum_{i=1}^{N_s} X_i = \nu\right)} \\
&= \prod_{u=1}^{N_s-1} \binom{N_c}{x_u} \left(\frac{1}{B}\right)^{x_u} \left(1 - \frac{1}{B}\right)^{N_c - x_u} \\
&\times \binom{N_c}{\nu - \sum_{i=1}^{N_s-1} x_i} \left(\frac{1}{B}\right)^{\nu - \sum_{i=1}^{N_s-1} x_i} \left(1 - \frac{1}{B}\right)^{N_c - \nu + \sum_{i=1}^{N_s-1} x_i} \\
&\times \left(\binom{N_c N_s}{\nu} \left(\frac{1}{B}\right)^{\nu} \left(1 - \frac{1}{B}\right)^{N_c N_s - \nu}\right)^{-1} \\
&= \prod_{u=1}^{N_s-1} \binom{N_c}{x_u} \binom{N_c}{\nu - \sum_{i=1}^{N_s-1} x_i} \binom{N_c N_s}{\nu}^{-1}.
\end{aligned} \tag{3.9}$$

So given ν, X is following multivariate hypergeometric distribution. Now, the probability $\mathbb{P}(1s|\nu, \mathbf{x} = \mathbf{x}')$ can be written as

$$\mathbb{P}(1s|\nu, \mathbf{x} = \mathbf{x}') = \sum_{m=1}^{N_s} \frac{x_m (x_m - 1)}{\nu (\nu - 1)}, \tag{3.10}$$

from which we can find $\mathbb{P}(1s|\nu)$ as follows

$$\mathbb{P}(1s|\nu) = \mathbb{E}_{\mathbf{x}|\nu} \left(\sum_{m=1}^{N_s} \frac{x_m (x_m - 1)}{\nu (\nu - 1)} \right) = \sum_{m=1}^{N_s} \mathbb{E}_{\mathbf{x}|\nu} \left(\frac{x_m (x_m - 1)}{\nu (\nu - 1)} \right)$$

$$\begin{aligned} \mathbb{E}_{\mathbf{x}|\nu} \left(\frac{x_m (x_m - 1)}{\nu (\nu - 1)} \right) &= \sum_{x_u=0}^{N_c} \frac{x_m (x_m - 1)}{\nu (\nu - 1)} \prod_{u=1}^{N_s - 1} \binom{N_c}{x_u} \binom{N_c}{\nu - \sum_{i=1}^{N_s - 1} x_i} \binom{N_c N_s}{\nu}^{-1} \\ &= \frac{N_c - 1}{N_s (N_c N_s - 1)} \end{aligned} \tag{3.11}$$

$$\Rightarrow \mathbb{P}(1s) = \sum_{i=1}^{N_s} \frac{N_c - 1}{N_s (N_c N_s - 1)} = \frac{N_c - 1}{(N_c N_s - 1)} \tag{3.12}$$

Next, we need to find $\mathbb{P}\left(\overline{C1}|1s\right) = 1 - \mathbb{P}(C1|1s)$ where, $\overline{C1} = \{f_{k_1} = f_{k_2}\}$ given the two vertices are induced by the same server.

Let $\mathbf{Z}_k$ be a random vector representing the number of clients requesting file f_k from the same server; where $\mathbf{Z}_k \sim Bin(N_c, P_s P_w)$, $P_w = \frac{1}{N_f}$ (each client wants only one file), and the total number of vertices induced by server s_i can be expressed as $X_i = \sum_{k=1}^{S} z_k$. By following the same steps in (3.9) we get

$$\mathbb{P}\left(\mathbf{z} = \mathbf{z}' | \sum_{k=1}^{S} Z_k = X_i \right) = \prod_{u=1}^{S-1} \binom{N_c}{z_u} \binom{N_c}{X_i - \sum_{v=1}^{S-1} z_v} \binom{N_c S}{X_i}^{-1}, \tag{3.13}$$

$$\mathbb{P}\left(\overline{C1}|\mathbf{z}=\mathbf{z}',\mathbf{x}=\mathbf{x}',1s\right)=\sum_{i=1}^{N_s}\sum_{k=1}^{S}\frac{z_k\left(z_k-1\right)}{x_i\left(x_i-1\right)},$$

$$\Rightarrow\mathbb{P}\left(\overline{C1}|1s\right)=\sum_{i=1}^{N_s}\sum_{k=1}^{S}\mathbb{E}_{\mathbf{x}|\nu}\left(\mathbb{E}_{\mathbf{z}|x_i}\left(\frac{z_k\left(z_k-1\right)}{x_i\left(x_i-1\right)}\right)\right),$$

$$\begin{aligned}\mathbb{E}_{\mathbf{z}|x_i}\left(\frac{z_k\left(z_k-1\right)}{x_i\left(x_i-1\right)}\right)&=\sum_{z_u=0}^{N_c}\frac{z_k\left(z_k-1\right)}{x_i\left(x_i-1\right)}\prod_{u=1}^{S-1}\binom{N_c}{z_u}\binom{N_c}{X_i-\sum_{v=1}^{S-1}z_v}\binom{N_cS}{X_i}^{-1}\\&=\frac{N_c-1}{S\left(N_cS-1\right)}.\end{aligned}$$

Since the resulting term does not depend on x, averaging over its distribution would result in the same term. Therefore

$$\mathbb{P}\left(\overline{C1}|1s\right)=\sum_{k=1}^{S}\frac{N_c-1}{S\left(N_cS-1\right)}=\frac{N_c-1}{N_cS-1},$$

$$\Rightarrow\mathbb{P}\left(C1|1s\right)=1-\frac{N_c-1}{N_cS-1}.\tag{3.14}$$

The next step is to find $\mathbb{P}\left(C2|1s,C1\right)=1-\mathbb{P}\left(\overline{C2}|1s,C1\right)$, where $\overline{C2}=\{f_{k_1}\in\mathcal{H}_{c_{j_2}}$ AND $f_{k_2}\in\mathcal{H}_{c_{j_1}}\}$. Based on the definition of the conflict-free IDNC graph, it is intuitive to say that $f_{k_1}\notin\mathcal{H}_{c_{j_1}}$ AND $f_{k_2}\notin\mathcal{H}_{c_{j_2}}$. Thus,

$$\mathbb{P}\left(\overline{C2}|1s, C1\right) = \mathbb{P}\left(f_{k_1} \in \left(\mathcal{H}_{c_{j_2}} \cap \mathcal{H}_{s_i}\right) \cap f_{k_2} \in \left(\mathcal{H}_{c_{j_1}} \cap \mathcal{H}_{s_i}\right)\right)$$
$$= \mathbb{P}\left(f_{k_1} \in \left(\mathcal{H}_{c_{j_2}} \cap \mathcal{H}_{s_i}\right)\right) \mathbb{P}\left(f_{k_2} \in \left(\mathcal{H}_{c_{j_1}} \cap \mathcal{H}_{s_i}\right) | f_{k_1} \in \left(\mathcal{H}_{c_{j_2}} \cap \mathcal{H}_{s_i}\right)\right). \tag{3.15}$$

To find $\mathbb{P}\left(\overline{C2}|1s, C1\right)$, we need to find the intersection between the has set of s_i and the has set of each client after removing the intersection between them. Let Y be a random variable denoting the number of files in the set $\mathcal{H}_{c_{j_1}} \cap \mathcal{H}_{c_{j_2}}$, which can be modelled as follows

$$\mathbb{P}\left(Y = y\right) = \binom{H}{y}\binom{N_f - H}{H - y}\binom{N_f}{H}^{-1}. \tag{3.16}$$

Let e be the number of the files in the set $\{\left(\mathcal{H}_{c_{j_1}} - \mathcal{H}_{c_{j_2}}\right) \cap \mathcal{H}_{s_i}\}$, which is modelled as

$$\mathbb{P}\left(e = e_1\right) = \binom{S}{e_1}\binom{N_f - S}{H - y - e_1}\binom{N_f}{H - y}^{-1}. \tag{3.17}$$

Let e' be the number of files in the set $\{\mathcal{H}_{s_i} - \{\left(\mathcal{H}_{c_{j_1}} - \mathcal{H}_{c_{j_2}}\right) \cap \mathcal{H}_{s_i}\}\} \cap \left(\mathcal{H}_{c_{j_2}} - \mathcal{H}_{c_{j_1}}\right)$, which is modelled as

$$\mathbb{P}\left(e' = e_2\right) = \binom{S - e_1}{e_2}\binom{N_f - (S - e_1)}{H - y - e_2}\binom{N_f}{H - y}^{-1}. \tag{3.18}$$

Using the three variables defined above, $\psi = \mathbb{P}\left(\overline{C2}|1s, C1\right)$ can be expressed as

$$\psi = \mathbb{P}\left(\overline{C2}|1s, C1\right) = \sum_{y=0}^{H} \frac{\binom{H}{y}\binom{N_f-H}{H-y}}{\binom{N_f}{H}} \sum_{e_1=max\left(0,H-\left(N_f-S\right)\right)}^{min(H-\mathbf{y},S)} \binom{S}{e_1}\binom{N_f-S}{H-y-e_1}\binom{N_f}{H-y}^{-1}\frac{e_1}{S}$$
$$\sum_{e_2=max\left(0,H-\left(N_f-(S-e_1)\right)\right)}^{min(H-\mathbf{y},(S-e_1))} \binom{S-e_1}{e_2}\binom{N_f-(S-e_1)}{H-y-e_1}\binom{N_f}{H-y}^{-1}\frac{e_2}{S-1}. \tag{3.19}$$

Hence, $\mathbb{P}\left(C2|1s, C1\right)$ is written as

$$\mathbb{P}\left(C2|1s, C1\right) = 1 - \psi. \tag{3.20}$$

By substituting (3.12),(3.14) and (3.20) in (3.8) we get

$$\mathbb{P}\left(E1\right) = \left(\frac{N_c - 1}{(N_c N_s - 1)}\right)\left(1 - \frac{(N_c - 1)}{N_c S - 1}\right)(1 - \psi) \tag{3.21}$$

To find $\mathbb{P}\left(E2\right) = \{c_{j_1} = c_{j_2} \text{ AND } s_{i_1} \neq s_{i_2}\}$, we need to recall that this event happens if a given client wants a file stored in two different servers. Therefore

$$\mathbb{P}\left(E2\right) = \mathbb{P}\left(f_k \in \mathcal{H}_{s_{i_1}} \; AND \; f_k \in \mathcal{H}_{s_{i_2}}\right) = \frac{R}{\nu}\frac{R-1}{\nu-1}. \tag{3.22}$$

From (3.21), (3.22) and (3.7) we get

$$\pi = \left(\frac{N_c - 1}{(N_c N_s - 1)}\right)\left(1 - \frac{(N_c - 1)}{N_c S - 1}\right)(1 - \psi) + \frac{R\left(R-1\right)}{\nu\left(\nu-1\right)}$$

Theorem 3.2 *The performance of the conflict-free IDNC algorithm is upper bounded by* $\chi(\mathcal{G}_{\nu,\pi})$.

proof: For graph $\mathcal{G}$, the chromatic number is the smallest required number of colors to color the vertices of $\mathcal{G}$ such that any two adjacent vertices do not share the same color [34]. Thus, the chromatic number represents the number of independent sets in that graph. In the case of one requested file per client, the transmission of an IDNC file will not generate any new coding opportunities since each client wants only one file and it will leave the competition once it receives the file. However, the dual conflict IDNC graph consists of $N_c R$ vertices. N_c of these vertices represent the actual number of the wanted files. The remaining $N_c(R-1)$ vertices represent repeated copies of the wanted files due to the file repetition among the servers. Since, after a request is being served, all the vertices representing this request will be removed from the graph after applying the conflict free IDNC algorithm. On the other hand, the chromatic number of the graph is calculated with presence of these repeated vertices and also the conflict edges connect them. Consequently, the actual number of transmissions should be smaller than the dual-conflict graph chromatic number. Hence, the performance of the conflict free IDNC algorithm is upper bounded by the chromatic number of the dual-conflict graph.

Corollary 1 *The performance of the dual conflict IDNC algorithm is lower bounded by* $\frac{\chi(\mathcal{G}_{\nu,\pi})}{R}$.

proof Let N^{IDNC} be the actual minimum number of the dual-conflict graph maximal independent sets after removing the copies. We need to prove the following

$$N^{IDNC} R \geq \chi\left(\mathcal{G}_{\nu,\pi}\right). \tag{3.23}$$

To do so, we need to use the original graph $\mathcal{G}_{\nu,\pi}$ to generate a new graph with a chromatic number $N^{IDNC}R$. This can be easily done by taking the graph containing the actual solution (the one with the minimum number of maximal independent sets) after removing all the copies, and repeating this graph R times. The last step is to connect all the R copies of that graph (i.e., every vertex in a copy of the graph should be connected to every vertex in another copy) to make sure that the chromatic number of the whole resultant graph is equal to the sum of the chromatic numbers of all copies.

The resultant graph will have a greater number of edges than the original one, because we do not connect a copy of a request to all the other copies of another request in $\mathcal{G}_{\nu,\pi}$. Since increasing the number of edges in a graph means it has a greater or equal chromatic number, the new graph chromatic number is greater than or equal to that of $\mathcal{G}_{\nu,\pi}$, and hence inequality in (3.23) is hold [37]. The theorem follows from dividing both sides in (3.23) by R.

3.6 Bron-Kerbosch Algorithm

Bron-Kerbosch algorithm was designed to find all the possible maximal cliques [1] in an undirected and unweighted graph. In other words, it lists all the possible subsets of vertices with two indispensable conditions. The first condition is that each two vertices in one of the listed subsets is adjacent by an undirected edge. The second condition is that each subset cannot be embedded by adding one more vertex [41]. The algorithm executes recursively as shown in Algorithm 3.1 and at any given point in running time the algorithm maintains three lists, κ, $\mathcal{G}^*$ and X. The set κ contains the vertices of the clique currently being calculated (κ is a possible maximal clique). The set $\mathcal{G}^*$ contains vertices that are connected to all vertices in κ and can be included to κ to make a larger clique. The set X contains vertices that are adjacent to all vertices in κ but are excluded from being added to κ because all cliques containing vertices in X have already been calculated in a previous recursion iteration.

Algorithm 3.1 BronKerbosch Algorithm - *Bron-Kerbosch*

1: Initialization:
   - Graph $\mathcal{G}^* \leftarrow \mathcal{G}$.
   - $\kappa = \phi$, $X = \phi$.
2: BronKerbosch(κ, $\mathcal{G}^*$, X):
3: **if** $\mathcal{G}^*$ and X are both empty,
4: **then**, report κ as a maximal clique.
5: **end if**
6: **for** each vertex v in $\mathcal{G}^*$ **do:**
7: BronKerbosch($\kappa \bigcup v$, $\mathcal{G}^* \bigcap \mathcal{N}(v), X \bigcap \mathcal{N}(v)$).
8: $\mathcal{G}^* \leftarrow \mathcal{G}^* \setminus v$.
9: $X \leftarrow X \bigcup v$.

[1] maximum clique is that clique which no more vertices can be included to it in. The clique with the largest size is the maxium clique [34].

The algorithm in 3.1 can be textualized as follows: the algorithm begins by choosing a vertex v randomly from $\mathcal{G}^*$. Add v to κ and remove its non-neighbors from $\mathcal{G}^*$ and X. Then pick another vertex from the new $\mathcal{G}^*$ set and repeat the process. The algorithm continues until $\mathcal{G}^*$ is empty. Once $\mathcal{G}^*$ is empty, if X is empty then report the content of κ as a new maximal clique (if it's not empty, then κ contains a subset of an already found clique). Now the algorithm will backtrack to the last vertex picked and restore $\mathcal{G}^*$, κ and X as they were before the choice. Finally, it will eliminate the vertex from $\mathcal{G}^*$ and include it to X, then expand the next vertex [2].

The optimal maximum clique in a graph can be found straightly using Bron-Kerbosch algorithm. Since the algorithm lists all the maximal cliques, the maximum clique is the largest one of them. However, to find the maximum independent set in a graph $\mathcal{G}$, we run the algorithm on the complement graph [3] $\overline{\mathcal{G}}$. In spite of the fact that the resultant (clique or maximum independent set) is optimum Bron-Kerbosch algorithm lists all the maximal cliques in the graph. In a graph with large number of vertices and edges, the implementation of Bron-Kerbosch algorithm is intractable due to the extremely large number of operations required to list all the maximal cliques.

[2]MATLAB Implementation of this version of Bron-Kerbosch algorithm is available at http://www.mathworks.com.

[3]The complement graph $\overline{\mathcal{G}}$ is a graph has the same vertices set as $\mathcal{G}$ and any two vertices are connected if and only if they are not connected in $\mathcal{G}$.[34].

3.6.1 Complexity Comparison with the Maximum Weighted Vertex Search Algorithm

In this section, we will compare the worst case complexity of the proposed maximum weighted vertex search algorithm discussed in Sec. 2.3.2 to that of the Bron-Kerbosch algorithm. In [42], it was proven that the worst-case time complexity of Bron-Kerbosch algorithm is given by $O(3^{\frac{n}{3}})$ for a graph with n vertices. The vertex set size of the dual conflict IDNC graph is $O(N_f N_c N_s)$. Consequently, the worst-case time complexity of Bron-Kerbosch algorithm for the dual conflict graph is given by $O(3^{\frac{N_f N_c N_s}{3}})$.

According to the description of the maximum weighted vertex search algorithm, the generation of one coded file requires $O(N_c)$ iterations. Since, any maximal independent set in the dual conflict IDNC graph cannot be of size larger than N_c (we can target each client at most once per transmission). Each iteration in the algorithm requires weight recomputations for the $O(N_f N_c N_s)$ graph vertices, the complexity of the algorithm is $O(N_f N_c^2 N_s)$.

3.7 Simulation Results

This section provides a comparison between the performance of the heuristic algorithm proposed in Sec. 2.3.2 and the optimum algorithm based on Bron-Kerbosch algorithm illustrated in Sec. 3.6. The proposed asymptotic upper and lower bounds derived in Sec. 3.5 are included in the simulation. Two cases are considered in the simulation, namely; fixed file placement and random file placement. In the former case, each file repeats exactly R times and the files are sorted in order in each server. In the later case, each server stores a set of files randomly. The only condition that governs the file distribution process is that all the files are stored in the servers collectively (i.e., $\bigcup_{i=1}^{N_s} \mathcal{H}_{s_i} = \mathcal{F}$).

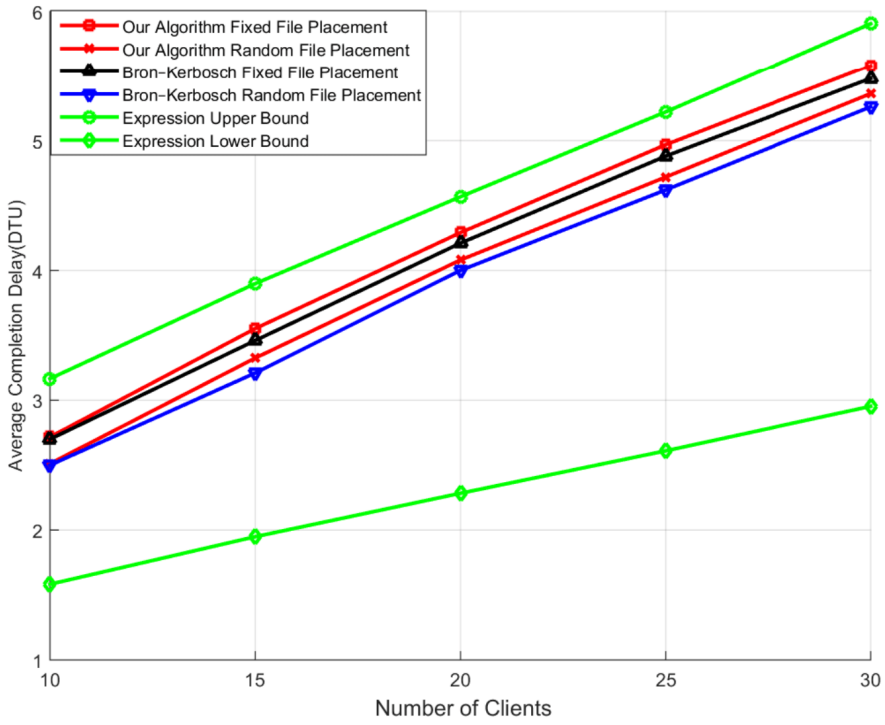


Figure 3.1: Performances Comparsion versus the number of the clients N_c.

Fig. 3.1 depicts the performance of the algorithms versus the number of clients with a fixed number of files $N_f = 100$. The number of servers is $N_s = 4$ and the size of each server is $\frac{N_f}{2}$. We assume corruption-free channels and each client already has $\lceil \frac{N_f}{3} \rceil$ files and wants to download only one file (i.e., $|\mathcal{H}_{c_j}| = \lceil \frac{N_f}{3} \rceil$ and $|\mathcal{W}_{c_j}| = 1$).

The performances of the algorithms versus the number of files, the fixed numbers of clients $N_c = 20$) is depicted in Fig. 3.2. We note that the heuristic algorithm performance is slightly close to the optimum solution provided by using Bron-Kerbuch algorithm. The effect of the file placement at the servers is notable and the performance of the random file placement is superior by about 0.2 DTU.

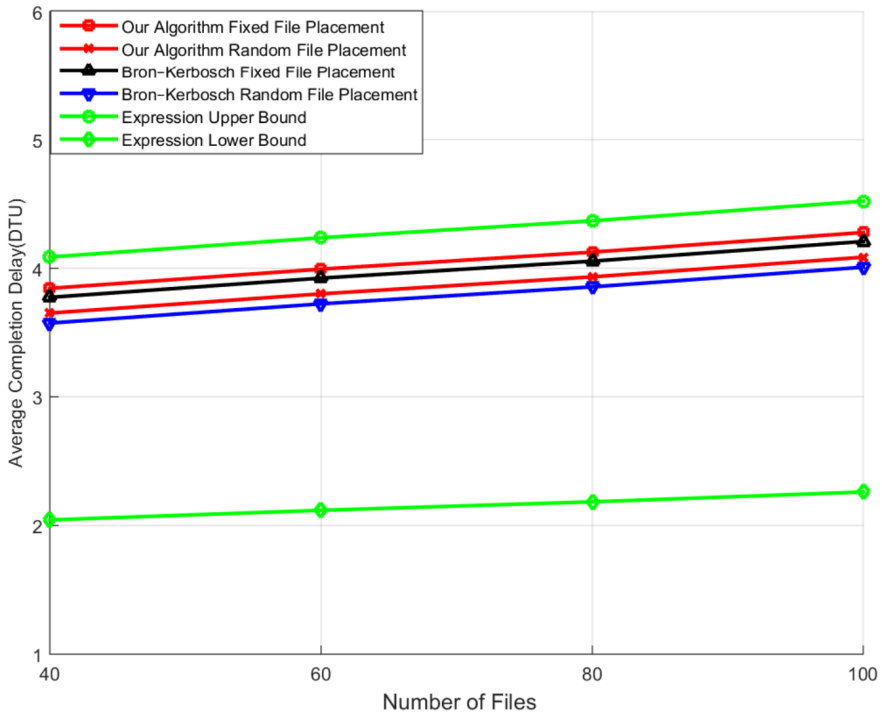


Figure 3.2: Performances Comparsion versus the number of files N_f.

3.8 Summary and Conclusions

In this chapter, upper and lower bounds of the conflict-free IDNC algorithm performance are derived. The upper bound was derived based on the fact the repeated vertices will increase the chromatic number of the dual conflict algorithm. On the other hand, these repeated vertices can not double the chromatic number by R, so the result of dividing the chromatic number by R is always less than the performance of the conflict-free IDNC algorithm. The bounds are valid for the case that one file is wanted by each client. In the derivations, we assumed fixed file placement in the server. However, the simulation results show that the bounds are still valid for the random file placement too. The optimum solution of the completion delay problem (based on Bron-Kerbosch algorithm) is compared with the proposed heuristic algorithm. we note from the resulted figures that the performance of the proposed heuristic algorithm is slightly close to the optimum one. In the next chapter, we will investigate the lossy feedback environment and its effect on the conflict-free algorithm. Multiple transmission schemes will be studied beside the maximum likelihood rule. We will investigate the centralized and the distributed implementation scenarios and study the different transmission schemes in each scenario.

CHAPTER 4

CONFLICT-FREE IDNC WITH LOSSY FEEDBACK

4.1 Motivation

In the second chapter, the reverse channels assumed to be perfect feedback channels. However the channel impairments will occur on both the forward and reverse links. In addition to that, the reverse link suffers from transmission power limitations and interference with other clients. Consequently, the feedback from the clients to the servers are subject to erasure. We refer to such imperfect reverse channels as Lossy Feedback (LF) channels. The difference between perfect and lossy feedback environments is the uncertainties introduced at the server due to unheard feedback occurrences. In perfect feedback environment, an unheard feedback event at the server makes the server certain that the transmitted file was lost on the client's forward link. However, the lossy feedback environment adds the

other possibility of file reception on the client's forward link and the loss of the feedback on its reverse link. Each of these possibilities happens independently for each of the targeted clients without feedback [22]. In that case, the server cannot determine the exact state of the network and thus cannot accurately select the subsequent IDNC file. The lossy feedback environment was considered as a marginal simulation setting in [43], [44] and [22]. In the first work [43], a guessing approach was used, in which a coin flip approach was adopted to decide on whether a file received at the client correctly or not. In [44], all uncertain files are temporarily ignored until further feedback is obtained about them. Partially blind selections of IDNC file combinations to minimize the IDNC completion in PMP system was adopted in [22]. In this chapter we will investigate the lossy feedback environment in a multipoint-to-multipoint systems and its effect on the completion delay problem in such systems. The remaining of this chapter is divided as follows. Section 4.2 investigates the feedback model in lossy feedback environment and the required adaptations on the augmented log matrix. In section 4.3, the completion delay problem in multipoint-to-multipoint system with imperfect feedback environment is formulated as a Partially-Observable SSP problem. The proposed completion delay minimization polices based on the Maximum Likelihood (ML) rules for both centralized and distributed implementation scenarios are illustrated in section 4.4. A partially blind graph update algorithm based on ML rules is discussed in section 4.5. Finally, the simulation results are illustrated in section 4.6.

4.2 Feedback Model

At the first time epoch, each server has full knowledge about the clients side information and the want set of each client. The feedback from clients assist the servers to update the graph based on the clients' want and has sets. Each targeted client c_j broadcasts a feedback packet to all servers after each download epoch. However, these feedback packets are themselves subject to erasure on the up-link channels. We will refer to the file with lost feedback at the server by the uncertain file of the client that sends the feedback. This adds an additional state to the log matrix to represent the uncertainty. Each server s_i can formulate its log matrix $L(s_i)$ as follows

$$L(s_i) = [l_{j,k}] \ \forall \ c_j \in \mathcal{C} \text{ and } f_k \in \mathcal{H}_{s_i},$$

$$s.t. \quad l_{j,k} = \begin{cases} 0, & c_j \text{ already has } f_k \\ 1, & c_j \text{ wants } f_k \\ -1, & c_j \text{ does not interested in } f_k \\ \text{x}, & s_i \text{ lose the feedback from } c_j \text{ about } f_k \end{cases} . \tag{4.1}$$

4.3 Lossy Feedback Problem Formulation

The difference between lossy and perfect feedback environments is the uncertainty of the server about the status of the transmitted file due to unheard feedback. In perfect feedback environment, an unheard feedback event at the server from a targeted client makes the server certain that the transmitted file is lost and the targeted client has not received it. However, the lossy feedback environment adds the possibility of file reception at the client and the loss of feedback at the server. In this case, the server cannot determine the exact log matrix and thus cannot accurately select the subsequent IDNC conflict-free file. These unheard feedback packets hide the exact state of the system from the server and convert the SSP formulation of the perfect feedback environment into a partially observable SSP (POSSP) problem [45]. The solution that was proposed in section 2.3 to solve the SSP problem can be adapted to solve the POSSP problem, if we can find a good estimate of the log matrix (and thus the IDNC conflict-free graph). In such stochastic partially observable problem, the best state estimate of the log matrix and the IDNC conflict-free graph is the one representing the maximum likelihood (ML) state of the system (i.e., the system state that has the highest probability) as well be discussed in the next section.

4.4 Maximum Likelihood (ML) State

The centralized and distributed implementation scenarios of the conflict-free algorithm induce disparity in the uncertainty state that may occurs at server s_i. In the former scenario, s_i will be uncertain about the file f_k reception at client c_j if all the servers in the system lose the feedback. Otherwise, the cloud unit will assist the uncertain servers to know the exact state. In the later scenario, s_i will be uncertain about the file f_k reception at a targeted client c_j if the server s_i target the client c_j , however it lose the feedback. This disparity in the uncertainty state required different maximum likelihood state estimation rules. The maximum likelihood state estimation rule will assist the server or the cloud unit to estimate the actual state of the transmitted file at the targeted client, either the file was received at the targeted client or lost in the forward sub-channel between the server s_i and the client c_j. In the centralized scenario, the decision will be unified and all the server will consider the uncertain file either received at the targeted client or not. Consequently, the transmission pattern still conflict-free even though the presence of the unheard feedback events. On the other hand, the distributed implementation allows each server to do its own decision while the other servers may be have the exact state of the file. This paradigm leads to diversity on the augmented graph structure and some conflicts may be slightly involved. The maximum likelihood state estimation rules are derived in the following two sections.

4.4.1 Algorithm Distributed Implementation Scenario

In the distributed scenario, the uncertainty state may occurs at server s_i as a result of one of the two following events:

- The client c_j is targeted by server s_i. However, c_j did not receive the file and thus did not emit a feedback packet. This case can happens with fixed probability p_{ij}.

- The client c_j is targeted by server s_i and c_j received the file and set a feedback, but the server could not receive it. This event occurs with probability $(1 - p_{ij})q_{ij}$.

This paradigm makes the distributed scenario to be identical to the PMP scenario. In [22] the ML rule for the PMP scenario was proved. In this work, we adopt the same ML rule which will be summarized as follows. For any client c_j, if any file is attempted n times by the server s_i, the server can set this file as not received at that client c_j if

$$\left(\frac{p_{ij}}{p_{ij} + (1 - p_{ij})\, q_{ij}}\right)^n \geq \frac{1}{2}. \tag{4.2}$$

In case of $n = 1$, the decision rule becomes

$$p_{ij} \geq (1 - p_{ij})\, q_{ij}.$$

4.4.2 Algorithm Centralized Implementation Scenario

The cloud unit will be uncertain about file f_k reception as a result of one of the two following events:

- The client c_j is targeted by any server s_i. However, c_j did not receive the file and thus did not emit a feedback packet. This case happens with fixed probability p_{ij}.
- The client c_j is targeted by any server s_i and c_j received the file and sent a feedback, but all the server could not receive it. This event occurs with probability $(1 - p_{ij}) \prod_{i=1}^{N_s} q_{ij}$.

Note that these two events are independent from each other.

Theorem 4.1 *For any client c_j, if any file is attempted n times, the cloud unit can set this file as not received at that client if*

$$\prod_{i=1}^{N_s} p_{ij}^{(n)_{ijk}} \geq \sum_{m=0}^{n-1} \binom{n}{m} \prod_{i=1}^{N_s} p_{ij}^{(m)_{ijk}} \prod_{i=1}^{N_s} (1 - p_{ij})^{(n-m)_{ijk}} \left(\prod_{i=1}^{N_s} q_{ij} \right)^{n-m}, \tag{4.3}$$

where $(n)_{ijk}$ is the number of attempts done by server s_i (out of the n attempts) *to target client c_j with file f_k.*

Proof: To prove this theorem, we define $\mathcal{P}_{jk}^{R}$ as the probability that an uncertain file f_k of client c_j has been received, and $\mathcal{P}_{jk}^{L}$ as the probability that an uncertain file f_k of client c_j has not been received. We need to derive expressions for $\mathcal{P}_{jk}^{L}$ and $\mathcal{P}_{jk}^{R}$. Let n be the total number of attempts the cloud unit scheduled a server to target client c_j by the file f_k without feedback. Each time the cloud unit schedules the same server or another server to target client c_j by the file f_k so let's define $(n)_{ijk}$ as the number of attempts (out of n attempts) to target client c_j by server s_i with file f_k ($n = \sum_{i=1}^{N_s} (n)_{ijk}$). Given these definition of n and n_{ijk}, the probability that file f_k is actually not received at client c_j is equal to the probability that this file was lost on the forward channels for each of the n attempts. This happens with probability $\prod_{i=1}^{N_s} p_{ij}^{(n)_{ijk}}$. On the other hand, the uncertain file f_k is actually received by client c_j if this client has successfully decoded it in any of the n attempts. However, the cloud unit lose the feedback(s). This occurs with probability $\sum_{m=0}^{n-1} \binom{n}{m} \prod_{i=1}^{N_s} p_{ij}^{(m)_{ijk}} \prod_{i=1}^{N_s} (1-p_{ij})^{(n-m)_{ijk}} \left(\prod_{i=1}^{N_s} q_{ij} \right)^{n-m}$. Therefore, given n attempts to transmit file f_k to client c_j with no feedback on any of them, $\mathcal{P}_{jk}^{L}$ and $\mathcal{P}_{jk}^{R}$ can be expressed as

$$\mathcal{P}_{jk}^{L} = \frac{\prod_{i=1}^{N_s} p_{ij}^{(n)_{ijk}}}{\prod_{i=1}^{N_s} p_{ij}^{(n)_{ijk}} + \sum_{m=0}^{n-1} \binom{n}{m} \prod_{i=1}^{N_s} p_{ij}^{(m)_{ijk}} \prod_{i=1}^{N_s} (1-p_{ij})^{(n-m)_{ijk}} \left(\prod_{i=1}^{N_s} q_{ij} \right)^{n-m}}, \quad (4.4)$$

$$\mathcal{P}_{jk}^{R} = \frac{\sum_{m=0}^{n-1} \binom{n}{m} \prod_{i=1}^{N_s} p_{ij}^{(m)_{ijk}} \prod_{i=1}^{N_s} (1-p_{ij})^{(n-m)_{ijk}} \left(\prod_{i=1}^{N_s} q_{ij}\right)^{n-m}}{\prod_{i=1}^{N_s} p_{ij}^{(n)_{ijk}} + \sum_{m=0}^{n-1} \binom{n}{m} \prod_{i=1}^{N_s} p_{ij}^{(m)_{ijk}} \prod_{i=1}^{N_s} (1-p_{ij})^{(n-m)_{ijk}} \left(\prod_{i=1}^{N_s} q_{ij}\right)^{n-m}} \tag{4.5}$$

$$= 1 - \mathcal{P}_{jk}^{L}.$$

The cloud unit can set the uncertain file f_k as not received at client c_j if

$$\mathcal{P}_{jk}^{L} \geq \mathcal{P}_{jk}^{R} \Rightarrow \mathcal{P}_{jk}^{L} \geq 1 - \mathcal{P}_{jk}^{L} \Rightarrow \mathcal{P}_{jk}^{L} \geq \frac{1}{2}$$

$$\Rightarrow \frac{\prod_{i=1}^{N_s} p_{ij}^{(n)_{ijk}}}{\prod_{i=1}^{N_s} p_{ij}^{(n)_{ijk}} + \sum_{m=0}^{n-1} \binom{n}{m} \prod_{i=1}^{N_s} p_{ij}^{(m)_{ijk}} \prod_{i=1}^{N_s} (1-p_{ij})^{(n-m)_{ijk}} \left(\prod_{i=1}^{N_s} q_{ij}\right)^{n-m}} \geq \frac{1}{2}$$

$$\Rightarrow \prod_{i=1}^{N_s} p_{ij}^{(n)_{ijk}} \geq \sum_{m=0}^{n-1} \binom{n}{m} \prod_{i=1}^{N_s} p_{ij}^{(m)_{ijk}} \prod_{i=1}^{N_s} (1-p_{ij})^{(n-m)_{ijk}} \left(\prod_{i=1}^{N_s} q_{ij}\right)^{n-m}.$$

In case of $n = 1$, the decision rule becomes

$$p_{ij} \geq (1-p_{ij}) \prod_{i=1}^{N_s} q_{ij}.$$

4.5 Partially Blind Graph Update Algorithm Based on ML

Based on the ML decision rules, we can adopt the partially blind graph update algorithm to solve the completion delay problem in lossy feedback environment proposed in [22]. When the server has uncertain files in its log matrix, due to unheard feedback, it can employ the above ML state estimation rules to update the conflict-free IDNC graph as follows. If the file f_k inducing the vertex $v_{i,j,k}$ is most likely received at client c_j (according to the ML rule), vertex $v_{i,j,k}$ is made hidden inside the conflict-free IDNC graph. This means that it is temporarily not considered for transmission. Otherwise, it is kept in the graph as an active vertex considered for any subsequent transmission.

The hidden vertices of any given client are treated according to what happens later:

- If the server receives a feedback from this client, it knows its actual state and can update the status of these hidden vertices.
- If a client does not have active vertices in the graph but only hidden ones, all these vertices are reset as active vertices and are re-transmitted within the subsequent conflict-free IDNC files until a feedback is received from this client.

4.6 Simulation Results

In this section, we consider the system described in chapter 2 with lossy feedback channels. Feedback loss would result in different download patterns for the centralized and the distributed scenarios described in Sec. 2.3.3.

In each simulation, we consider the conventional PMP IDNC and the dual-conflict IDNC algorithm implemented in the centralized and the distributed scenarios with lossy feedback environment. We assume that $q_{ij} = \frac{1}{2}p_{ij}$ and p_{ij} is uniformly distributed over the interval $[0.01, 0.3]$ uniformly. The ML rule in (4.2) is also applied in the conventional PMP IDNC. In addition to the partially blind algorithm based on ML described in section 4.4, two other blind graph update approaches are simulated:

- Full Vertex Elimination (FVE): Each attempted vertex with unheard feedback is assumed to be hidden in the graph and all its repeated copies are temporarily not considered for transmission.

- Client Block (CB): Each attempted vertex with unheard feedback causes all the vertices representing files wanted by client c_j and all the repeated copies of these files to be hidden in the graph and are temporarily not considered for transmission. These hidden vertices are treated later as described in the algorithm in Section 4.5.

Figs. 4.1-4.3 depict the performance of the different graph update approaches implemented for the dual conflict IDNC (in the centralized and the distributed scenarios) and the conventional IDNC verses N_c. With ($N_f = 100$, server size of 50 files), N_f ($N_c = 60$, server size of $\frac{1}{2}N_f$) and the server size S ($N_s = 10$, $N_f = 100$), respectively. The same performance verses the ratio $\frac{q_{ij}}{p_{ij}}$ (with $N_c = 60$, $N_f = 100$ and server size of 50) shown in Fig. 4.4. For this figure, p_{ij} is constant and q_{ij} is varying from $0.2p_{ij}$ up to the worst case $q_{ij} = p_{ij}$. From these figures, we can

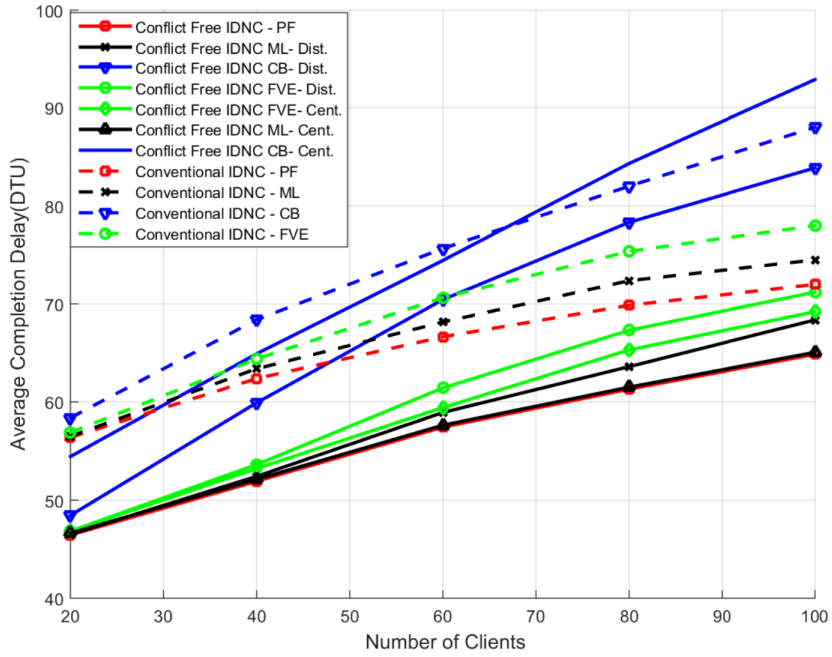


Figure 4.1: The average completion delay versus the number of the clients N_c.

observe the superiority in the performance of the ML approach over the FVE and the CB in reducing the average completion delay. For the centralized scenario, the performance of the ML is slightly close to that of the perfect feedback. Due to the fact that the lose of the feedback in the centralized scenario resulted from the feedback channels is very rare event. This makes the ML rule decision more

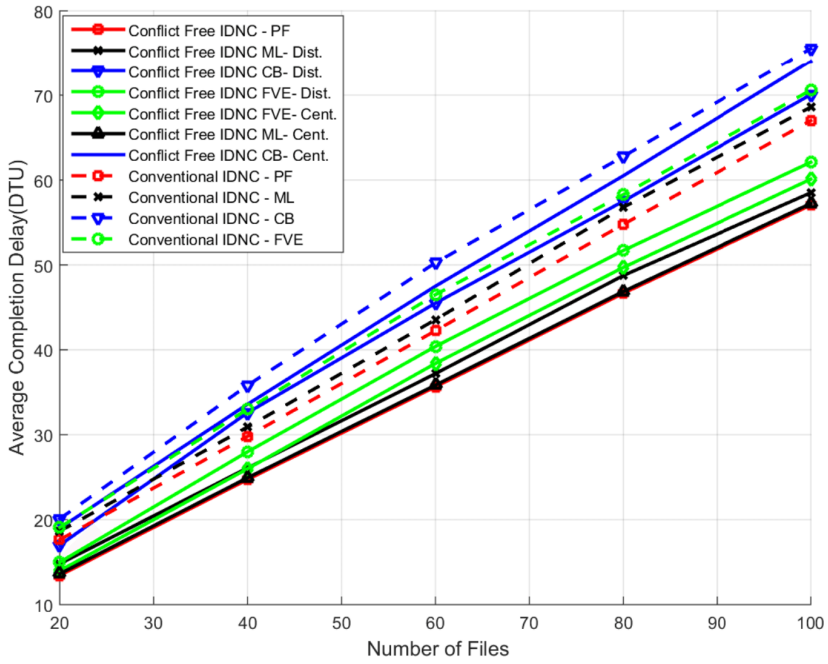


Figure 4.2: The average completion delay versus the number of the files N_f.

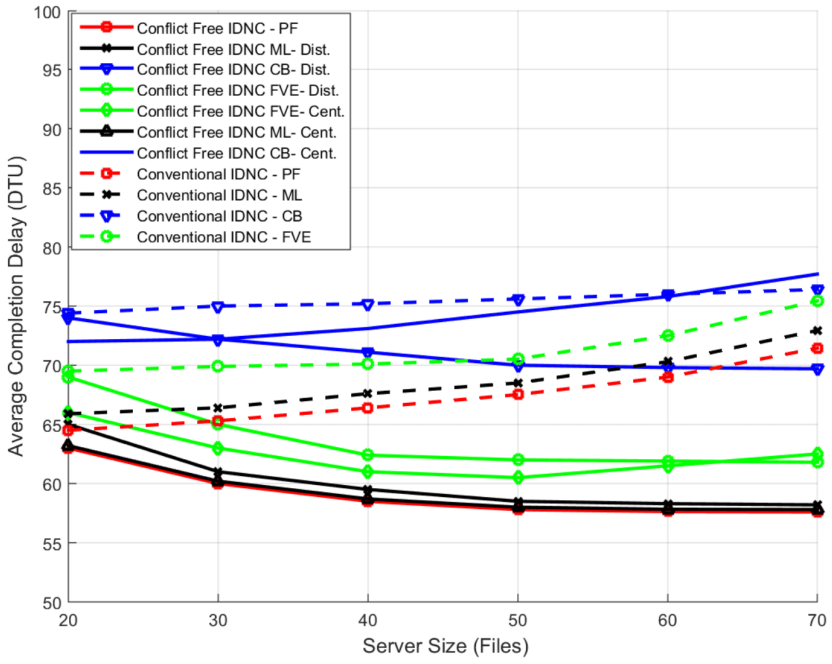


Figure 4.3: Perfect and Lossy feedback performance comparison over a range of the server size S.

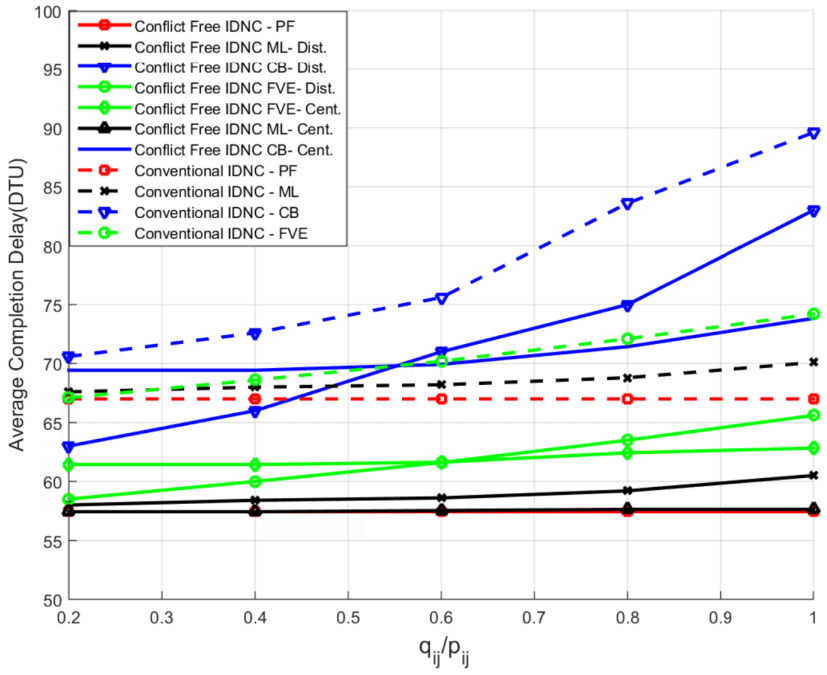


Figure 4.4: Perfect and Lossy feedback performance comparison over a range of the ratio $\frac{q_{ij}}{p_{ij}}$.

accurate. The results show that the ML update approach of the graph represents the ML estimation of the actual graph. We can also observe that FVE outperforms CB. This can be deduced from the characteristics of the two approaches. FVE hides the vertex v_{ijk} which represent unheard feedback and its repeated copies. However, the server s_i is still able to target the client c_j with another wanted file. On the other hand, CB hides the vertex v_{ijk} and all the vertices induced by client c_j. Consequently, the server s_i will block client c_j. Another factor added to the CB in the centralized scenario is that once a client is blocked the system will not target that client until one of the servers has no active vertices. We can also notice that the difference in performance between the dual conflict IDNC and the conventional IDNC are still noticeable in lossy feedback environment.

4.7 Summary and Conclusions

In this chapter, the completion delay reduction problem for conflict-free IDNC with imperfect (lossy) feedback is considered. The problem is first formulated as a partially observable SSP problem. This problem has the same structure and properties as that of the perfect feedback environment problem. Also the formulated POSSP problem has the considerably large size and solution intractability as the former SSP problem. Consequently, we proposed maximum likelihood estimation rule to estimate the state of the transmitted file without feedback. Which aid the severs to do accurate vertex and graph update decisions . The results indicated that the the derived maximum likelihood algorithm achieved a remarkable reduction in the completion delay in the distributed scenario. It also achieved near optimal performance in the centralized scenario. The full vertex elimination approach performs better than client block approach in the centralized and the distributed scenarios. Our algorithm ensures conflict-free IDNC transmissions with the lossy feedback environment in the centralized scenario. However, the distributed scenario may involved some intermediate conflicts.

In the next chapter, we conclude this thesis work and summarize the main contributions and results. We also propose some ideas as future works to be considered as extensions to this work.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this work, the minimization problem of the download time in DSNs using IDNC with perfect and lossy feedback environments is considered. Applying the conventional PMP IDNC algorithm is shown to result in multiple transmissions conflicts, which reduce the downloading efficiency. Consequently, we proposed a novel graph model which provides conflict-free transmissions. The problem was first formulated as an SSP problem which required a heuristic algorithm. This algorithm utilizes a maximum weight vertex search scheme over the dual-conflict graph to find the most suitable files combination at each time epoch. The lower and upper bounds of the conflict-free IDNC algorithm performance were derived. The simulation results show that this bounds are valid for both the

random files placement and the fixed files placement in the servers. The proposed heuristic algorithm shows near-optimum performance in the random and fixed files placement in the servers compared to the optimum solution acquired using BronKerbosch algorithm.

We extended the SSP formulation in perfect feedback to a POSSP formulation reflecting the uncertainties resulting from unheard feedback events in the lossy feedback environment. Then, we used this formulation to identify the ML state of the system in unheard feedback events, and employed it to design a partially blind dual conflict graph update. In addition to the partially blind algorithm based on ML, we simulated two other blind graph update approaches, namely; the full vertex elimination approach and the client block approach. Simulation results show that the proposed conflict-free IDNC scheduling outperformed the conventional IDNC in both perfect and lossy feedback environments.

5.2 Future Work

There are many open research problems for conflict-free IDNC field that need to be investigated and evaluated under different system parameters and new designs have to be proposed. Here we are enumerating some of these areas of research as follows:

Conflict-Free Data Exchange Systems : data exchange system is a set of communicating nodes at which the nodes are transceiver and each node stores a portion of the files library and missed the others. This model is also suitable for communication networks with content cashing from the base stations. After the first shoot (transmission) from the base station, some files are erased duo to the channel impairments. Assuming all the file library was scattered among a set of the nodes after the first transmission, however each node still missing some files. A cooperative data exchange can be unleashed among the aforementioned nodes set to complete the missed files without the need for re-transmission from the transmitted base station. This will free the transmitted base station to serve another set of the nodes. The conflict-free IDNC algorithm can be implemented in the cooperative data exchange system. At each time epoch, a subset of the nodes involving into the data exchange framework can be selected to be local transmitters. The conflict-free IDNC algorithm can be upgraded to be able to select this temporary transmitters and schedule the targeted nodes by each transmitting node with conflict-free IDNC file combinations.

Adaptive file size IDNC transmissions : The essential condition of the IDNC network coding is the equally sized files. If the file size varies from one client to another, the implementation of the IDNC will be unattainable. However, our dual conflict IDNC graph model can be developed to enable the transmitter to deal with a groups of users each group shares the same file size. The transmitter can represent each wanted file by a vertex. In addition to the IDNC conflicts among the vertices induced by files with the same size, any two vertices induced by files with different size should be set connected by an undirected edge to indicate that these two files can not be encoded together. In that case, the problem will be graph partitioning problem instead of the problem of finding the maximum weighted independent set in the dual-conflict graph.

REFERENCES

[1] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.

[2] D. S. Papailiopoulos, J. Luo, A. G. Dimakis, C. Huang, and J. Li, "Simple regenerating codes: Network coding for cloud storage," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2801–2805.

[3] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure-coding based routing for opportunistic networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. ACM, 2005, pp. 229–236.

[4] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE/ACM Trans. on Networking (TON)*, vol. 14, no. SI, pp. 2809–2816, 2006.

[5] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth codes: Maximizing sensor network data persistence," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4. ACM, 2006, pp. 255–266.

[6] M. S. ElBamby, M. Bennis, W. Saad, and M. Latva-aho, "Content-aware user clustering and caching in wireless small cell networks," in *Wireless Commun. Systems (ISWCS), 2014 11th International Symposium on.* IEEE, 2014, pp. 945–949.

[7] N. A. P. S. Yousef Shnaiwer, Sameh Sorour and T. Al-Naffouri, "Network-coded content delivery in femtocaching-assisted cellular networks," in *accepted for presentation and publication in the IEEE Globecom 2015*, 2015.

[8] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network inf. flow," *Inf. Theory, IEEE Trans. on*, vol. 46, no. 4, pp. 1204–1216, 2000.

[9] J. Huang, L. Wang, W. Cheng, and H. Li, "Polynomial time construction algorithm of bcnc for network coding in cyclic networks," in *Computer and Inf. Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conf. on.* IEEE, 2009, pp. 228–233.

[10] S. Sorour and S. Valaee, "On densifying coding opportunities in instantly decodable network coding graphs," in *Inf. Theory Proceedings (ISIT), 2012 IEEE International Symposium on.* IEEE, 2012, pp. 2456–2460.

[11] U. J. Ferner, P. Sadeghi, N. Aboutorab, and M. Medard, "Scheduling advantages of network coded storage in point-to-multipoint networks," in *Network Coding (NetCod), 2014 International Symposium on.* IEEE, 2014, pp. 1–6.

[12] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proceedings of the Annual Allerton Conf. on Communi-*

cation Control and Computing, vol. 41, no. 1. The University; 1998, 2003, pp. 11–20.

[13] L. Junhai, Y. Danxia, X. Liu, and F. Mingyu, "A survey of multicast routing protocols for mobile ad-hoc networks," *Commun. Surveys & Tutorials, IEEE*, vol. 11, no. 1, pp. 78–91, 2009.

[14] S. K. D. K. W. Hu and H. R. M. Médard, "The importance of being opportunistic: Practical network coding for wireless environments," *Newsletter ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, 2006.

[15] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," *IEEE/ACM Trans. on Networking (ToN)*, vol. 16, no. 3, pp. 497–510, 2008.

[16] A. Le, A. S. Tehrani, A. G. Dimakis, and A. Markopoulou, "Instantly decodable network codes for real-time applications," in *Network Coding (NetCod), 2013 International Symposium on*. IEEE, 2013, pp. 1–6.

[17] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *Veh. Tech., IEEE Trans. on*, vol. 58, no. 2, pp. 914–925, 2009.

[18] P. Sadeghi, D. Traskov, and R. Koetter, "Adaptive network coding for broadcast channels," in *Proc. 5th Workshop on Network Coding, Theory and Applications*, 2009, pp. 80–86.

[19] P. Sadeghi, R. Shams, and D. Traskov, "An optimal adaptive network coding scheme for minimizing decoding delay in broadcast erasure channels," *EURASIP Journal on Wireless Commun. and Networking*, vol. 2010, p. 4, 2010.

[20] S. Sorour and S. Valaee, "Completion delay minimization for instantly decodable network codes," *Networking, IEEE/ACM Trans. on*, vol. PP, no. 99, pp. 1–1, 2014.

[21] ——, "On minimizing broadcast completion delay for instantly decodable network coding," in *Commun. (ICC), 2010 IEEE International Conf. on.* IEEE, 2010, pp. 1–5.

[22] S. Sorour, A. Douik, S. Valaee, T. Y. Al-Naffouri, and M.-S. Alouini, "Partially blind instantly decodable network codes for lossy feedback environment," *Wireless Commun., IEEE Trans. on*, vol. 13, no. 9, pp. 4871–4883, 2014.

[23] A. Douik, S. Sorour, M.-S. Alouini, and T. Y. Al-Naffouri, "Delay reduction in lossy intermittent feedback for generalized instantly decodable network coding," in *Wireless and Mobile Computing, Networking and Commun. (WiMob), 2013 IEEE 9th International Conf. on.* IEEE, 2013, pp. 388–393.

[24] R. W. Watson, "High performance storage system scalability: Architecture, implementation and experience," in *Mass Storage Systems and Technologies,*

2005. Proceedings. 22nd IEEE/13th NASA Goddard Conf. on. IEEE, 2005, pp. 145–159.

[25] Q. Lian, W. Chen, and Z. Zhang, "On the impact of replica placement to the reliability of distributed brick storage systems," in *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conf. on.* IEEE, 2005, pp. 187–196.

[26] B. Lampson and H. Sturgis, *Crash recovery in a distributed data storage system.* Xerox Palo Alto Research Center Palo Alto, California, 1979.

[27] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[28] J. Tate, P. Beck, H. H. Ibarra, S. Kumaravel, L. Miklas *et al.*, *Introduction to storage area networks and system networking.* IBM Redbooks, 2012.

[29] H. Yamamoto and D. Maruta, "Replication methods for load balancing on distributed storages in p2p networks," *IEICE Trans. on Inf. and systems*, vol. 89, no. 1, pp. 171–180, 2006.

[30] D. A. Patterson, G. Gibson, and R. H. Katz, *A case for redundant arrays of inexpensive disks (RAID).* ACM, 1988, vol. 17, no. 3.

[31] E. Erez and M. Feder, "Efficient network code design for cyclic networks," *Inf. Theory, IEEE Trans. on*, vol. 56, no. 8, pp. 3862–3878, 2010.

[32] M. A. R. Chaudhry and A. Sprintson, "Efficient algorithms for index coding," in *INFOCOM Workshops 2008, IEEE*. IEEE, 2008, pp. 1–4.

[33] S. Y. El Rouayheb, M. A. R. Chaudhry, and A. Sprintson, "On the minimum number of transmissions in single-hop wireless coding networks," in *Inf. Theory Workshop, 2007. ITW'07. IEEE*. IEEE, 2007, pp. 120–125.

[34] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[35] A. A. Al-Habob, S. Sorour, N. Aboutorab, and P. Sadeghi, "Conflict free network coding for distributed storage networks," in *Commun. (ICC), 2015 IEEE International Conf. on*. IEEE, 2015, pp. 5517–5522.

[36] P. ERDdS and A. R&WI, "On random graphs i," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[37] B. Bollobás, "The chromatic number of random graphs," *Combinatorica*, vol. 8, no. 1, pp. 49–55, 1988.

[38] E. N. Gilbert, "Random graphs," *The Annals of Mathematical Statistics*, pp. 1141–1144, 1959.

[39] R. Van Der Hofstad, "Random graphs and complex networks," *Available on http://www. win. tue. nl/rhofstad/NotesRGCN. pdf*, 2009.

[40] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[41] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Commun. of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[42] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theoretical Computer Science*, vol. 363, no. 1, pp. 28–42, 2006.

[43] R. Bruno and M. Nurchis, "Survey on diversity-based routing in wireless mesh networks: Challenges and solutions," *Computer Commun.*, vol. 33, no. 3, pp. 269–282, 2010.

[44] E. Magli, M. Wang, P. Frossard, and A. Markopoulou, "Network coding meets multimedia: A review," *Multimedia, IEEE Trans. on*, vol. 15, no. 5, pp. 1195–1212, 2013.

[45] S. D. Patek, "On partially observed stochastic shortest path problems," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conf. on*, vol. 5. IEEE, 2001, pp. 5050–5055.

Vitae

- Name: Ahmed Abdullah Ali Al-habob
- Nationality: Yemeni
- Date of Birth: 02/03/1984
- Email: *alhaboob2011@outlook.com*
- Permenant Address: Taiz, Yemen

- MSc. Degree in Telecommunication Engineering, KFUPM, Dhahran, KSA, 2015.
- BACHELOR'S DEGREE in Telecommunication and Computer Engineering (Telecommunication Sector), Taiz UNIVERSITY, 2009.
- Research field of interest: Network Coding, cooperative networks, cognitive radio and Wireless communications.
- Publications:
  - [**C1**]: A. A. Al-Habob, S. Sorour, N. Aboutorab, and P. Sadeghi, "Conflict free network coding for distributed storage networks," in *Commun. (ICC), 2015 IEEE International Conf. on.* IEEE, 2015, pp. 55175522.